НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації і управління

«На правах рукопису»

УДК _004.65:004.457___

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Магістерська дисертація

зі спеціальності *121 «Інженерія програмного забезпечення»*

на тему: «Математичне та програмне забезпечення обробки надвеликих масивів даних у форматі XML»

Виконав: студент VI курсу, групи *ІП-81мп*

Педоренко Олег Русланович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *ст. викл. Олійник Ю. О.* _____
(посада, науковий ступінь, , вчене звання, прізвище та ініціали) (підпис)

Консультант *доц., к.т.н., Ліщук К.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. АУТС, к.т.н. Буксов М. М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *121 «Інженерія програмного забезпечення»*
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
«___» _____ 20*1* р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Педоренко Олегу Руслановичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Математичне та програмне забезпечення обробки надвеликих масивів даних у форматі XML

науковий керівник дисертації *ст. викл. Олійник Ю. О.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "____" _____ 20__ р. № _____

2. Строк подання студентом дисертації "____" *грудня* 20*19* р.

3. Об'єкт дослідження Надвеликі масиви даних у форматі XML

4. Предмет дослідження Методи обробки надвеликих масивів даних у форматі XML

5. Перелік завдань, які потрібно розробити *1. Визначити потребу в обробці надвеликих масивів даних у форматі XML; 2. Аналіз наявних методів і засобів*

обробки XML; 3. Розробити модель обробки XML-даних та надвеликих масивів таких даних; 4. Розробити метод перетворення даних з умовою масового паралельного виконання; 5. розробити архітектуру програмного забезпечення; 6. реалізувати програмне забезпечення; 7. дослідити ефективність розробленого методу.

6. Перелік графічного матеріалу

1. *Схема розгортання компонентів програмного забезпечення*
2. *Алгоритм обробки надвеликих масивів XML даних*

7. Орієнтовний перелік публікацій *1 публікація*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Графічний	Ліщук К. І., доц.		

9. Дата видачі завдання “ *01* ” *вересня* 20 *19* р

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Аналіз існуючих методів обробки XML даних*	*15.09.2019*	
2	*Порівняльний аналіз існуючих методів обробки XML даних*	*25.09.2019*	
3	*Постановка математичної моделі задачі*	*01.10.2019*	
4	*Розробка математичного забезпечення*	*10.10.2019*	
5	*Розробка інформаційного та програмного забезпечення*	*21.10.2019*	
7	*Проведення експериментальних досліджень розробленого алгоритму*	*10.11.2019*	
8	*Оформлення документації*	*29.11.2019*	
9	*Подання роботи на попередній захист*	*05.12.2019*	
10	*Подання роботи на основний захист*	*16.12.2019*	

Студент _____ Педоренко О. Р.
(підпис) (ініціали, прізвище)

Науковий керівник _____ Олійник Ю. О.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми: XML – це популярний формат для передачі і зберігання даних. На цьом форматі побудовані стандарти для обміну даними у багатьох галузях діяльності. Тема роботи є актуальною, оскільки на сьогодні існуючі засоби роботи з XML не дають можливості аналітичної обробки надвеликих масивів даних.

Мета дослідження: розробка засобів аналітичної обробки надвеликих масивів XML-документів, що добре інтегруються з існуючими системами зберігання даних

Для реалізації поставленої мети були сформульовані **наступні завдання**:

- розробити модель обробки XML-даних та надвеликих масивів таких даних;
- розробити метод перетворення даних з умовою масового паралельного виконання;
- розробити архітектуру програмного забезпечення, що реалізовує такий метод;
- реалізувати програмне забезпечення для обробки надвеликих масивів XML-даних;
- дослідити ефективність розробленого методу.

Об'єкт дослідження: надвеликі масиви даних у форматі XML

Предмет дослідження:.методи обробки надвеликих масивів даних у форматі XML

Методи дослідження: при проведенні досліджень у дисертаційній роботі використовувались методи обробки надвеликих масивів даних на основі масових паралельних обчислень.

Наукова новизна: Найбільш суттєвими науковими результатами магістерської дисертації є:

– вперше створено метод обробки надвеликих масивів даних у форматі XML, що дозволяє виконання аналітичних запитів;

– розроблено програмне забезпечення, що використовує створений метод.

Практичне значення отриманих результатів визначається тим, що запропонований алгоритм багаторазово прискорює процес аналізу надвеликих масивів даних у форматі XML.

Зв'язок роботи з науковими програмами, планами, темами: дисертаційна робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Методи та технології високопродуктивних обчислень та обробки надвеликих масивів даних». Державний реєстраційний номер 0117U000924.

Апробація: Основні положення роботи доповідались і обговорювались на III Всеукраїнській науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) у рамках доповіді на тему «Метод обробки надвеликих масивів XML даних»

Ключові слова: XML, ВЕЛИКІ ДАНІ, MAP REDUCE, ПОВНОТЕКСТОВИЙ ПОШУК

ABSTRACT

Relevance of the topic: XML is a popular format for transmitting and storing data. This format builds standards for data exchange in many industries. The topic of work is relevant, because today the existing XML tools do not allow analytical processing of large data sets.

Research goal: development of analytical tools for processing large arrays of XML documents that integrate well with existing storage systems

The following **tasks** have been set to reach the goal:

- develop a model for processing XML data and large arrays of such data;
- to develop a method of data conversion with the condition of mass parallel execution;
- develop a software architecture that implements this method;
- implement software for processing oversized XML data sets;
- to investigate the effectiveness of the developed method.

Research object: extra-large XML data sets

Research subject: methods of processing large data sets in XML format

Research methods: the methods of processing of large data sets based on mass parallel calculations were used in the dissertation research.

Scientific novelty: The most significant scientific results of the master's thesis are:

– for the first time the method of processing of large data sets in XML format was created, which allows to perform analytical queries;
– software developed using the created method.

Practical value of the results: The practical value of the obtained results is determined by the fact that the proposed algorithm significantly improves the process of analysis of large data sets in XML format.

Link to scientific programs: This dissertation was performet at the Department of Automated Information Processing and Management Systems of the National Technical University of Ukraine "Kyiv Polytechnic Institute named after Igor Sikorsky" within the

topic “Methods and technologies of high-performance computing and processing of large data sets”. State Registration Number 0117U000924.

Publications: The main provisions of the work were reported and discussed at the III All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems and Technologies of Management" (ISTU-2019) in the framework of the report on the method of processing large data sets of XML data

Keywords: XML, BIG DATA, MAP REDUCE, FULL TEXT SEARCH

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

BLOB (Binary Large Object) – двійковий великий о’єкт

XML (eXtensible Markup Language) – розширювана мова розмітки

DTD (Document Type Definition) – визначення типів документів

XSD (XML Schema Definition) – визачння схеми XML

ПЗ – програмне забезпечення

СКБД – система керування базами даних

JSON (JavaScript Object Notation) – об’єктна нотація Javascript

SQL (Structured Query Language) – структурована мова запитів

ВСТУП

Обробка надвеликих масивів даних – це тренд останнього десятиліття. Зростання потужності комп'ютерів, пропускних здатностей мереж, здешевлення засобів зберігання інформації призвели до можливості накопичення великих об'ємів записів про роботу користувачів із відповідними системами. Провідні світові компанії зрозуміли, що обробка цієї часто неструктурованої, хаотичної інформації може надати доступ до неочевидних знань, які надають конкурентну перевагу – таких як розуміння шаблонів поведінки клієнтів, передбачення їх дій, тощо.

На жаль, старі технології роботи з інформацією не завжди оптимальним чином підходять для обробки даних. Так, зберігання XML документів у реляційній базі даних, що одного часу було непоганим способом обійти деякі обмеження цих двох прийомів, не підходить для агрегації та збору статистичної інформації про документи. Незважаючи на те, що такий прийом широко використовується у індустрії банківського, страхового, медичного програмного забезпечення для зберігання слабо типізованих масивів документів та транзакцій, досі немає способу ефективної обробки таких даних.

Сучасні технології дозволяють розділити зберігання даних для їх обробки з метою аналізу та з метою подальшої роботи з ними, що дозволить зберети зворотню сумісність та не зламати існуючі системи, надавши користувачам можливості глибокого дослідження того, що зберігається у них на серверах.

Метою дослідження є розробка засобів аналітичної обробки надвеликих масивів XML-документів, що добре інтегруються з існуючими системами зберігання даних.

1 ПРОБЛЕМА ОБРОБКИ НАДВЕЛИКИХ МАСИВІВ XML ДАНИХ

1.1 Загальні визначення і терміни

1.1.1 XML

XML є мовою розмітки, яка була створена для опису даних. Мова розмітки є набором символів або послідовностей, що вставляються в текст для передачі інформації про його виведення або будову. Таким чином, текстовий документ, розмічений за допомогою такої мови, містить не лише сам текст, але і додаткову інформацію про його структуру. Крім того, мова розмітки дозволяє вставляти в документ інтерактивні елементи і зміст інших документів. Розмітка розділяється на стилістичну розмітку, структурну і семантичну: стилістична розмітка відповідає за зовнішній вигляд документу, структурна розмітка задає структуру документу, семантична дозволяє описати логіку представлення даних. XML є підмножиною метамови SGML, розробленою для спрощення процесу машинного розбору документу. Можна сказати, що він сам є метамовою, оскільки не обмежується набором певних тегів і використовується як засіб для опису граматики інших мов і контролю за правильністю складання документів. XML-документ зазвичай складається з процесингових інструкцій, елементів, атрибутів, сутностей і коментарів. [1]

Приклад XML документа наведено у лістингу 1.1 [2].

Лістинг 1.1 – XML документ

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <bookstore>
3     <book category="COOKING">
4       <title lang="en">Everyday Italian</title>
5       <author>Giada De Laurentiis</author>
6       <year>2005</year>
7       <price>30.00</price>
8     </book>
9     <book category="CHILDREN">
1       <title lang="en">Harry Potter</title>
0
1       <author>J K. Rowling</author>
1
```

Лістинг 1.1 – XML документ

```
      <year>2005</year>

      <price>29.99</price>

    </book>

    <book category="WEB">

      <title lang="en">XQuery Kick Start</title>

      <author>James McGovern</author>

      <author>Per Bothner</author>

      <author>Kurt Cagle</author>

      <author>James Linn</author>

      <author>Vaidyanathan Nagarajan</author>

      <year>2003</year>

      <price>49.99</price>

    </book>

    <book category="WEB">

      <title lang="en">Learning XML</title>

      <author>Erik T. Ray</author>

      <year>2003</year>

      <price>39.95</price>

    </book>

  </bookstore>
```

Мова XML була створена для зберігання, транспортування і обміну даними, з її допомогою можна реалізувати обмін даними між різними системами. XML документ складається з частин, що називаються елементами. Елементи складають основу XML-документів. Вони утворюють структури, які можна обробляти програмно або за допомогою таблиць стилів. Елементи розмічають іменовані розділи інформації. Елементи будуються за допомогою тегів розмітки, що означають ім'я, початок і кінець елементу. Елементи можуть бути вкладені один в одного, на верхньому рівні

знаходиться елемент, що називається елементом документу або кореневим елементом в якому містяться інші елементи.

З точки зору своєї структури XML-документ є впорядкованим деревом, вершинами якого є елементи. Коренем дерева є кореневий елемент, листками дерева – дані, що містяться в елементах. На рисунку 1.1 зображено структуру документа, наведеного у лістингу 1.1 [2].

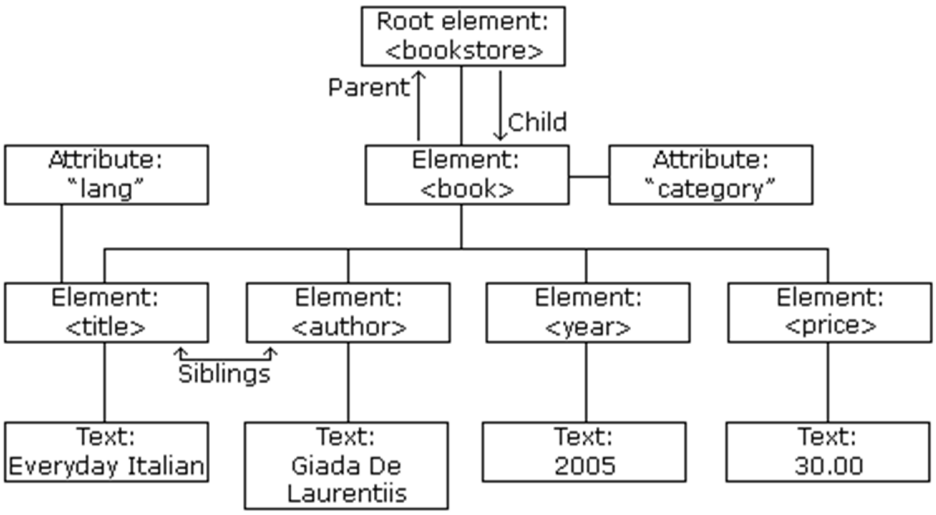


Рисунок 1.1 – Структура XML-документа як впорядковане дерево

1.1.2 Засоби опису структури XML документів

Існує декілька поширених засобів опису структури XML-документів. Найбільш поширеними є DTD та XML Schema, описані нижче.

DTD [1] (Document Type Declaration – опис типу документа) – це особливий елемент, складова XML документа, що описує його структуру. Може міститися в самому документі, або у зовнішньому файлі та підключатися до документа. DTD задає граматику XML документа:

- як можуть називатись елементи документа;
- в якій послідовності елементи мають бути розташовані;
- кількість повторів певних елементів в докуемнті;

– відношення між елементами (які інші елементи може містити певний елемент).

Завдяки опису DTD будь-який XML-документ можна перевірити на валідність, тобто на відповідність наперед заданій структурі. Це дозволяє автоматизовану обробку таких документів.

У лістингу 1.2 наведено приклад DTD, що описує документ із лістингу 1.1.

Лістинг 1.2 – DTD для XML документа

```
1     <!ELEMENT bookstore (book*)>
2
3     <!ELEMENT book (title, author, year, price)>
4     <!ELEMENT title (#PCDATA)>
5     <!ELEMENT author (#PCDATA)>
6     <!ELEMENT year (#PCDATA)>
7     <!ELEMENT price (#PCDATA)>
8
9     <!ATTLIST book category CDATA>
10    <!ATTLIST title lang CDATA>
```

XML Schema – це ще один спосіб описати структуру XML документа. Його також називають XML Schema Definition (XSD). XML Schema є більш потужною альтернативою DTD, і має ряд переваг [2]:

– XML Schema також є XML документом;
– можливість розширення документів XML Schema;
– підтримка типів даних, у тому числі можливість створення власних типів даних;
– можливість створення об’єктної моделі на основі XSD, що спрощує роботу з XML у об’єктно орієнтованих мовах програмування.

У лістингу 1.3 наведено приклад XSD, що описує документ із лістингу 1.1. Можна побачити, що такий опис є менш компактним, проте більш повним та структурованим.

Опис XML-документа у форматі XML Schema називають XSD-схемою [2] та зберігають у текстових файлах із розширенням **.xsd*.

Лістинг 1.3 – XSD для XML документа

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="title">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="lang"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="author" type="xs:string"/>
  <xs:element name="year" type="xs:short"/>
  <xs:element name="price" type="xs:float"/>
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="author"/>
        <xs:element ref="year"/>
        <xs:element ref="price"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="category"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="bookstore">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="book"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1.1.3 Запити та перетворення XML-документів

Для читання даних із XML-документів у програмних додатках існує декілька стандартизованих засобів: мова запитів: XPath, мова запитів і перетворень XQuery та мова перетворень XSLT.

XPath – це стандарт мови запитів до XML документів. Це мова виразів, що є основним інструментом роботи з такими документами та споживання даних, що містяться в таких докментах.

Найбільш використовуваний функціонал XPath – це вибір елементів XML та повернення їх значень. Також XPath передбачає можливість навігації по всьому дереву документа для вибору та агрегації значень елементів [3].

Приклад запиту до документу з лістингу 1.1: «*/bookstore/book[1]/title*». Результат такого запиту: «*Everyday Italian*»

XQuery – специфікація для мови запитів, яка дозволяє користувачеві або програмісту витягувати інформацію з файлу XML або будь-якої XML-подібної колекції даних. Синтаксис призначений для легкого розуміння та використання. Використовуючи XQuery, можна розглядати таблицю реляційної бази даних як документ XML. XQuery – це специфікація, що розвивається та розробляється організацією World Wide Web Consortium (W3C) і має широку підтримку декількох основних постачальників, включаючи IBM, Microsoft та Oracle.

XQuery використовує XPath – мову, що описує спосіб пошуку та обробки елементів у документах XML. Технічні характеристики XPath та XQuery тісно пов'язані. XQuery 1.0 походить безпосередньо від XPath 2.0. У XQuery вирази XPath можуть бути простими запитами або частинами великих запитів. XQuery має функції для численних операцій, включаючи порівняння дат і часу, математичні обчислення, рядкові маніпуляції та булеву алгебру. Якщо необхідної функції немає в XQuery, програміст може написати її власноруч. [15]

XQuery використовує функції для доступу до даних з XML документів. Нехай документ з лістингу 1.1 зберігається у файлі *books.xml*. Тоді результатом виконання запиту XQuery «*doc("books.xml")/bookstore/book/title*» буде набір елементів, зазначений у лістингу 1.4.

Лістинг 1.4 – Результат виконання запиту XQuery

```
1  <title lang="en">Everyday Italian</title>
2  <title lang="en">Harry Potter</title>
3  <title lang="en">XQuery Kick Start</title>
4  <title lang="en">Learning XML</title>
```

XSLT (eXtensible Stylesheet Language Transformations) – декларативна мова, що описує перетворення XML документів. Основна мета перетворень – сформувати структуру документа, відмінну від початкового документа, але із тими ж самими даними. Прикладами застосування може бути перетворення між різними XML Schema, або створення HTML-звітів на основі XML-документів.

1.2 Способи зберігання XML документів

Існують різні способи зберігання XML документів – такі як зберігання документів у файловій системі [5], реляційній базі даних [6, 7, 8], об’єктно-орієнтованому сховищі [9, 10] або XML-орієнтованій базі даних [16]. Розглянемо коротко кожен із них.

1.2.1 Текстовий файл

За цього способу кожен XML-документ зберігається як текстовий файл. Один із способів реалізації механізму запитів для роботи з даними при такому підході – це завантаження XML-файла в оперативну пам’ять у вигляді дерева, до якого потім виконується запит. Дерево зберігається в пам'яті доти, доки будь-які вузли в дерева потрібні для виконання запитів. За такого способу виявляється, що час розбору займає більшу частину часу виконання запитів до даних. Цей підхід є неприпустимо повільним. Можна створити спецаіальну систему запитів, що може використовувати індекси для отримання сегментів файлу XML, відповідних запиту, різко скорочуючи час аналізу. [4]

1.2.2 Реляційна база даних

Сучасні системи керування базами даних (СКБД), такі як Oracle Database [11], Microsoft SQL Server [12], MySQL [13] мають спеціальні типи даних для зберігання XML файлів, дозволяючи виконувати запити до цих документів у складі SQL-запитів. Також будь-яка СКБД дозволяє зберігати XML-документи у вигляді великого рядка

або BLOB. Недоліки використання реляційних баз даних для мети зберігання документів – принципова неможливість побудови будь-яких індексів по таким типам даних, необхідність обробки документа на стороні додатку, що споживає дані, а також неможливість виконання групових або агрегуючих запитів, що використовують дані з XML-документа. Оскільки для виконання кожного запиту СКБД мусть проводити повний розбір кожного документа, швидкість роботи такого методу є незадовільною. У такому сценарії реляційна база даних фактично використовується так само, як файлова система, тільки XML-документи зберігаються не у вигляді файлів, а у вигляді записів у реляційній таблиці.

Існують оптимізації вищенаведеного способу [6, 7, 8], що передбачають попередній розбір документа та його структури, і перетворення їх у набір таблиць реляційної бази даних. Це дозволяє не проводити розбір документа кожен раз при виконанні запиту і прибирає вищезгадані недоліки (неможливість побудови індексу і агрегації, тощо). [4] Втім, це вже не є зберіганням XML документа як такого. Сам документ необхідно перебудовувати кожен раз, коли він необхідний клієнту, що також має додану вартість порівняно із простим зберіганням цілого документа.

1.2.3 Об’єктно-орієнтоване сховище

В об’єктно-орієнтованому сховищі XML-дані зберігаються у вигляді ієрархічної структури об’єктів. В об’єктах зберігають дані про XML-елементи такі як дочірні і материнський елемент, дані та атрибути елемента. По дереву об’єктів будують індекс B-Tree [14], який можна використати для пошуку будь-якого елемента за його іменем та властивостями. У рамках цього підходу кожен раз при запиті цілісного документа його необхідно перебудовувати заново. Проте це набагато швидше, ніж перебудовування документа з таблиць реляційної бази даних. [4]

1.3 Проблема десеріалізації XML документів

Формат XML не є єдиним форматом зберігання і серіалізації даних. Існує також ряд інших форматів:

- CSV
- JSON
- Protobuf (Protocol Buffers)
- TOML
- YAML

Нижче розглянемо коротко кожен із них.

1.3.1 Формат CSV

CSV (Comma separated values – значення, розділені комою) добре підходить для зберігання великої кількості табличних даних у читабельному форматі. Він не підходить для зберігання об'єктів або хеш-таблиць, як структури даних (на відміну від усіх інших форматів серіалізації, які переглядаються тут).

CSV не дуже добре стандартизований. RFC 4180 була спробою стандартизації формату, однак назва "CSV" може посилатися на файли, які розмежовані символами без комами, такими як пробіли, вкладки або напівколонки. Насправді його раніше називали DSV (Delimiter separated values – значення, розділені обмежувачем), хоча, на жаль, CSV є більш поширеним терміном в ці дні. А формат даних між роздільниками повністю визначений користувачем (наприклад, немає визначеного способу визначення дат).

Формат CSV дозволяє необов'язковий рядок заголовка, який відображається як перший рядок у файлі. Якщо він присутній, він містить назви полів для кожного значення в записі. Цей заголовок дуже корисний для маркування даних і майже завжди повинен бути присутнім.

Формат CSV добре підтримується: бібліотеки CSV доступні майже для кожної популярної мови програмування. Популярні панди бібліотеки для обробки даних для Python можуть читати в CSV прямо в таблицю даних (називається Dataframe) за допомогою простої однорядкової команди pd.read_csv ('myfile.csv'). CSV - це справді єдиний формат серіалізації, який розглядається на цій сторінці, який має хорошу підтримку в програмах електронних таблиць, таких як Excel (це єдиний переглянутий формат серіалізації, який має сенс читати в Excel, оскільки CSV виконує табличну структуру).

CSV досить компактний. Однак може бути важко розрізнити стовпчики , особливо коли є велика кількість рядків (у верхній частині файлу є лише один рядок заголовка), є велика кількість стовпців (немає вимоги, щоб стовпці були однаково розташовані, і таким чином ви мусите підраховувати коми зліва).

Приклад файлу у форматі CSV наведено нижче.

Лістинг 1.5 – CSV документ

```
1    id, title, author, genre
2    4, Harry Potter and the Goblet of Fire, J. K. Rowling, Fantasy
3    25, A Song of Ice and Fire, Fantasy
```

Отже, основною перевагою CSV є його компактність, широка підтримка різними програмами та бібліотеками. Також він дуже швидкий при серіалізації та десеріалізації. Недоліком CSV є те, що він пристосований винятково до зберігання даних сталої табличної структури. [20]

1.3.2 JSON

JSON – це розповсюджений формат серіалізації даних, який підтримується майже всіма популярними мовами програмування. Структури даних JSON тісно представляють загальні об'єкти багатьох мов програмування, хоча існують винятки.

Важливо зазначити, що синтаксис JSON не підтримує коментарів. Що стосується великої кількості даних, які читаються в програмне забезпечення і

повертаються на диск, коментарі в будь-якому разі виявляться марними, оскільки вони не зберігаються при повторному записі у файл. Ім'я в парах імен / значень об'єкта JSON завжди повинно бути рядком. Він також не підтримує жодного типу формату дати.

Приклад JSON наведено нижче.

Лістинг 1.6 – JSON документ

```
[
    {
        "id": 4,
        "title": "The Lord of The Rings",
        "author": “J. R. R. Tolkien”,
        "genre": "Fantasy"
    },
    {
        "id": 4,
        "title": "The art of Clean Code",
        "author": “Jerry Mander”,
        "genre": "Technical literature"
    }
]
```

Серед переваг JSON – лаконічний синтаксис, добра підтримка популярними мовами програмування, дуже швидка серіалізація та десеріалізація. Недоліки – відсутність стандартної можливості валідації, довільний формат та структура даних (хоча це може бути й перевагою у деяких випадках). [20]

1.3.3 Protocol Buffers (Protobuf)

Protobuf - це бінарний протокол серіалізації, розроблений Google. Оскільки вона серіалізується у двійковій формі, вона не читається людиною (хоча ви все одно можете вибирати рядки під час перегляду файлу як текст ASCII, див. Приклад нижче).

Типи даних, які може містити Protobuf, чітко визначені і включають загальні типи, такі як рядки, цілі та дробові числа, та дати.

Лістинг 1.7 – JSON документ

```
#^R^Charmander^Z^Fire Street%<A2>@
^A^R^Pikachu^Z^Electric Street%<F1>'OA
```

Protobuf, будучи двійковим форматом, дуже стислий. Він також використовує бітове кодування для типів даних, таких як цілі числа, щоб зменшити кількість байтів, коли числа невеликі. Формат швидкий при серіалізації та десеріалізації, підтримується популярними мовами програмування. Також його використовують при обміні даними у мережі Інтернет для зменшення об’ємів трафіку, що передається. Недоліком формату є те, що він абсолютно не читається людиною і не має вбудованих засобів валідації даних. При використанні великої кількості рядкових даних переваги формату зменшуються, так як він зберігає рядки без змін. [20]

1.3.4 TOML

TOML (Tom’s Obvious Minimal Language) - це новіший (відносно інших у цьому списку) формат серіалізації, що читається людиною. Він досить схожий на YAML в тому, що спрямований на файли конфігурації, але прагне бути більш простим форматом (YAML може отримати досить складний, і це можна побачити у набагато повільніші часи розбору YAML).

TOML має підсвічувачі синтаксису для Atom, Visual Studio, Code Visual Studio та інших IDE.

TOML страждає від дещо багатослівного синтаксису, коли мова йде про вираження масиву об'єктів (у TOML говорити, масив таблиць). Це можна побачити на прикладі нижче, коли кожен об’єкт книги в масиві розмежовано символом [[book]].

Лістинг 1.8 – TOML документ

```
[[book]]
id = 4
title = "Charmander"
Author = "Fire Street"
genre = 12.34

[[book]]
id = 25
title = "Pikachu"
author = "Electric Street"
genre = 56.78
```

TOML – дуже схожий на JSON з точки зору переваг та недоліків. Цей формат добре підходить для застосування у файлах конфігурацій тощо. [20]

1.3.5 YAML

YAML – це ще один варіант JSON, що відрізняється від нього строгою специфікацією. Приклад документу YAML наведено нижче.

Лістинг 1.9 – YAML документ

```
1     - { id: 0, name: Charmander, age: 12.34, address: "Fire Street" }
2     - { id: 25, name: Pikachu, age: 56.78, address: "Electric Street" }
```

YAML має обширну специфікацію, що робить його певною мірою аналогом XML. Також ця мова підтримує різноманітні типи даних. Утім, через свою складну структуру ця мова розмітки не підходить до зберігання або передачі великих обсягів даних: швидкість серіалізації та десеріалізації є найповільнішою з-поміж інших альтернатив. [20]

1.4 Застосування надвеликих масивів XML даних

За останні десятиліття XML став стандартним форматом обміну даними у багатьох індустріях, зокрема у фінансовій індустрії. Незаважаючи на те, що останнім часом популярінсть XML стала зменшуватись, у користувачів накопичилися великі масиви даних у цьому форматі, які необхідно обробляти та аналізувати.

Нижче наведено приклади використання сховищ даних у форматі XML з реального життя.

1.4.1 Природничі науки

Компанія, що проводить дослідження в області природничих наук, постійно отрмує і створює XML документи розміром від 10 Мб до 500 Мб. Їх зберігають просто як файли у файловій системі. Компанії необхідно завантажити ці файли в реляційну, або нереляційну базу даних для їх обробки, отримання статистичних даних, звітів по

даним що містяться у файлах, тощо. Проте жодна існуюча система не може ефективно обробляти XML файли такого розміру. [17]

1.4.2 Великий банк

OLTP-система (OnLine Transaction Processing, обробка транзакцій в режимі реального часу) великого банку обслуговує більше 20 000 користувачів, виконуючи близько 17 мільйонів транзакцій на день. Файли транзакцій мають стандартизований XML-формат. [17]

1.4.3 Банківська система

У банківській системі, що обслуговує сотні клієнтів, використовується власний формат XML-документів. Усі транзакції для багатьох видів банківських продуктів, таких як цінні папери, іпотеки, споживчі та підприємницькі кредити зберігаються у реляційній базі даних у вигляді XML-файлів. Це дозволяє легко інтегрувати різні підсистеми, завдяки єдиному формату використовуваних даних. У системі зберігаються дані про мільйони транзакцій, що користувачі створили за декілька років.

Користувачі та розробники банківської системи бажають отримати доступ до аналітики по тим даним, що уже зберігаються в системі. Наприклад, мати можливість оцінити кількість виданих споживчих кредитів по місяцях року. Або знати кількість транзакцій, що використовують ті чи інші можливості системи для оцінки ризиків при зміні функціональності.

1.4.4 Цінні папери

Компанія є одним з найбільших світових постачальників IT-систем для обробки брокерських операцій та даних про цінні папери. Їх система використовується для обробки операцій з різними фінансовими інструментами, такими як акції та фонди, а також для управління рахунками клієнтів та фірм. Одна частина їх системи отримує

декілька потоків (10 і більше) даних про цінні папери. Ці потоки можуть бути або не бути у форматі XML, але їх потрібно перетворити у загальний формат XML, зберігати та запитувати в базі даних. [17]

1.5 Постановка задачі

Як було показано вище, проблема обробки надвеликих масивів XML-даних є актуальною. Для розв'язання проблеми необхідно:

- розробити модель обробки XML-даних та надвеликих масивів таких даних;
- розробити метод перетворення даних з умовою масового паралельного виконання;
- розробити архітектуру програмного забезпечення, що реалізовує такий метод;
- реалізувати програмне забезпечення для обробки надвеликих масивів XML-даних;
- дослідити ефективність розробленого методу.

1.6 Висновок

Проблема обробки надвеликих масивів XML-даних є досить поширеною. Джерелом проблеми є архітектура XML-файлів – їх деревоподбна структура, надлишковість, текстовий формат унеможливлюють створення індексів по XML-файлам у реляційних базах даних, а нереляційні бази даних надають перевагу зберіганню даних у інших форматах. Іще одним джерелом проблеми є кількість різних способів зберігання XML-документів, що утруднює створення універсального інструменту для аналізу надвеликих масивів XML-даних.

Було продемонстровано актуальність та необхідність розробки більш ефективного методу аналізу надвеликих масивів даних у форматі XML та поставлено задачі для досягнення цієї мети.

2 МОДЕЛІ ТА МЕТОДИ ОБРОБКИ МАСИВІВ XML ДАНИХ

2.1 Змістовна постановка задачі

У попередньому розділі ми розглянули різні способи зберігання XML даних. Ми показали, як ці способи дозволяють зберігати надвеликі масиви (десятки тисяч документів) XML-даних з різною ефективністю. Необхідно розробити метод зберігання та відтворення XML-даних, який би задовільняв наступним критеріям:

1) Можливість виконання запитів, в першу чергу агрегаційних та аналітичних запитів до даних;
2) можливість масштабувати виконання методу для обробки надвеликих масивів XML документів;
3) можливість завантаження даних до популярних систем обробки надвеликих масивів даних.

Розглянуті у попередньому розділі методи обробки XML документів не задовольняють цим вимогам. Вони здебільшого концентруються на побудові власної XML-орієнтованої бази даних, або на створенні синтезу XML та реляційної моделі зберігання даних.

Ми пропонуємо альтернативний метод: зберігання XML-даних у вигляді пар «ключ-значення». Такий вибір має декілька причин:

- Популярність систем для великих даних, що зберігають та обробляють дані саме таким чином (приклади - MongoDB, Redis, Elasticsearch, Hadoop), що означає потенційну можливість експорту даних до будь-якої з них і позбавляє необхідності створювати свою власну систему для обробки і візуалізації даних.
- Швидкість запитів у такій моделі зберігання даних, адже структура даних з деревоподібної стає лінійною.
- Можливість створення індексів за ключами для швидкого пошуку.

Єдиним недоліком подібної структури є відносна складність відновлення вхідного документа. Цю проблему можна розв'язати, зберігаючи документ у файловій системі або простому сховищі документів разом із обробленою структурою пар ключ-значення.

2.2 Математична постановка задачі

Представимо XML-документ у вигляді дерева:

$$G = (V, E), \tag{2.1}$$

де V – множина вершин дерева, E – множина його граней.

Кожній вершині дерева відповідає окремий елемент документа, кожній грані – зв'язок між материнським і дочірнім елементом документа.

Множиною значень Z назвемо множину листків графа G. Множиною шляхів Q назвемо таку множину, що містить всі шляхи від кореня дерева G до його листків.

Множину пар ключ-значення, таким чином, ми можемо визначити як множину пар

$$L = (Z, Q) \tag{2.2}$$

Необхіно описати функцію $f(G)$, таку що перетворює дерево XML-документа G у множину пар ключ-значення, L:

$$L = f(G) \tag{2.3}$$

2.3 Моделі зберігання надвеликих масивів XML даних

2.3.1 Структури даних для зберігання документа

Щоб робити запити до документу, зберігатимемо його як множину пар ключ-значення – використаємо структуру, що є списком таких пар. У якості ключа вкауватимемо XPath певного значення у документі (у якому символи "/" та "." замінимо на точки) (див. лістинг 2.1).

Лістинг 2.1 – Структура зберігання документу

```
1    { "rootElement.element1" : "value",
2      . . .
3    "rootElement.nestedElement.element2" : "value2" }
```

У якості значення – власне значення, що зберігається за деяким XPath. Усі XPath будуть абсолютними відносно кореневого елемента. На лістингу 2.2 показано, яку структуру матиме документ з лістингу 1.1

Лістинг 2.2 – Приклад перетвореного документу

```
{ "bookstore.book.0.@category" : "COOKING",
"bookstore.book.0.title.@lang" : "en",
"bookstore.book.0.title" : "Everyday Italian",
"bookstore.book.0.author" : "Giada De Laurentiis",
"bookstore.book.0.year" : "2005",
"bookstore.book.0.price" : "30.00",
"bookstore.book.1.@category" : "CHILDRE",
"bookstore.book.1.title.@lang" : "en",
"bookstore.book.1.title" : "Harry Potter",
"bookstore.book.1.author" : "J K. Rowling",
"bookstore.book.1.year" : "2005",
"bookstore.book.1.price" : "29.99",
"bookstore.book.2.@category" : "WEB",
"bookstore.book.2.title.@lang" : "en",
"bookstore.book.2.title" : "Learning XML",
"bookstore.book.2.author" : "Erik T. Ray",
"bookstore.book.2.year" : "2003",
"bookstore.book.2.price" : "39.95" }
```

2.4 Методи обробки масивів XML документів

2.4.1 Перетворення XML документів

Для перетворення документів із XML вигляду на список пар ключ-значення був розроблений наступний алгоритм, представлений у лістингу 2.3.

Лістинг 2.3 – Алгоритм перетворення документа

```
Вхідні дані:
  Дерево XML G(V, E)
Вихідні дані:
  Список пар ключ-значення L(Z, Q)
Алгоритм ПеретворитиНаПару:
  Визначити всі дочірні елементи поточної вершини;
  Якщо поточний елемент є листком, повернути пару ключ-значення, де ключ - значення поточного елементу, ключ - значення материнського елементу.
  Для кожного дочірнього елемента виконати поточний алгоритм.
  До списку ключів результату виконання попереднього кроку додати поточний елемент.
  Повернути результат
```

Цей алгоритм – це реалізація функції *f(G)*, заданої виразом (2.3). Алгоритм є рекурсивним. Для досягнення більшої ефективності роботи на багатопроцесорних системах, можна кожен рекурсивний виклик алгоритму виконувати в окремому потоці. Якщо є велика кількість документів, можна також обробляти кожен документ в окремому потоці, що прискорить час виконання алгоритму [22].

2.4.2 Паралельна обробка запитів

Щоб зробити алгоритм масштабованим на велику кількість серверів, що зберігають та обробляють дані, застосуємо концепцію розподіленої обробки даних MapReduce [18]. Запити до документів будуть оброблятись окремо на кожному сервері, де зберігаються документи, після чого результати запитів будуть підсумовуватись або оброблятись іншим чином для отримання єдиного значення.

На рисунку 2.1 зображено модель виконання задач Map Reduce. [19]

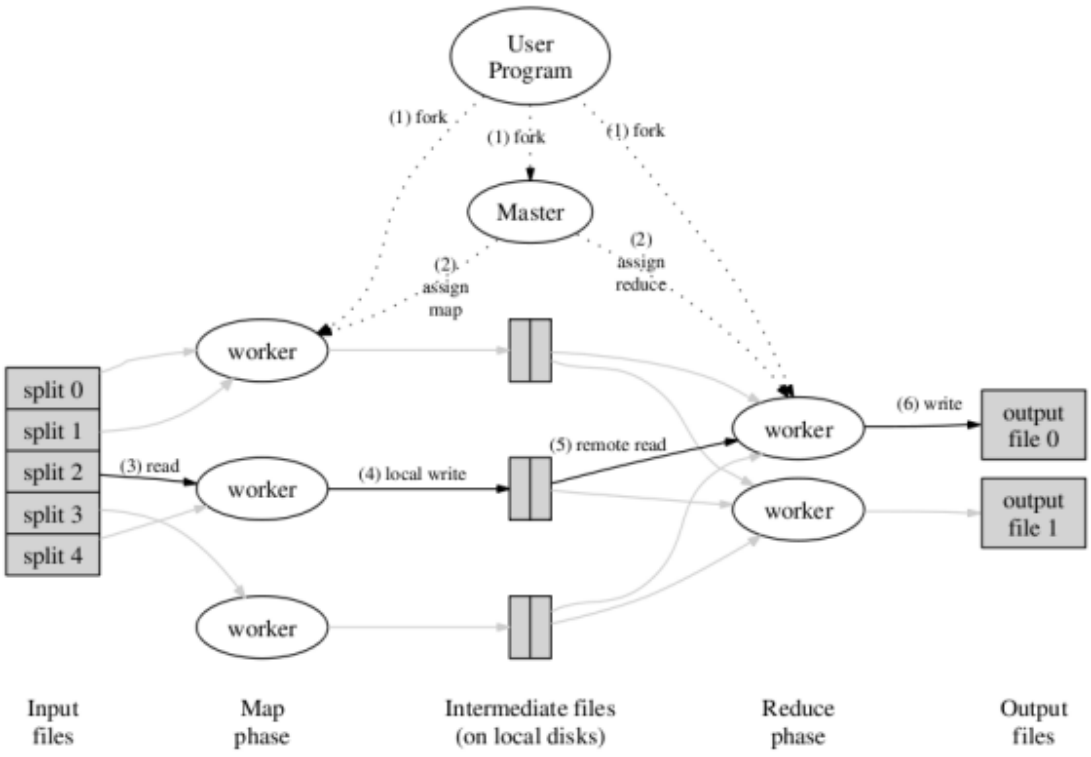


Рисунок 2.1 – Схема виконання задач Map/Reduce

Пояснимо термінологію, що стосується процесу:

- Машина (Machine) – фізичний комп'ютер, частина розподіленого кластеру;
- Задача (Task) – процес, що виконується на машині;
- Нода (Node) – виконавець процесу, в ідеальному випадку – один на машину.

Модель Map/Reduce можна коротко описати так [19]:

1. Фреймворк Map/Reduce спочатку розбиває файли вхідних даних на *M* фрагментів фіксованого розміру - це, як правило, від 16 до 64 мегабайт (МБ) за штуку (керується користувачем за допомогою додаткового параметра). Ці *M* фрагментів потім передаються машинам-учасникам кластеру. Зазвичай є 3 копії (керовані користувачем) кожного фрагменту з метою запобігання відмов. Потім він запускає багато копій програми користувача (в нашому випадку, запити до попередньо оброблених XML-файлів) на вузлах кластера.
2. Один з вузлів кластера особливий – головний (Master). Решта - це робітники (workers), яким призначено роботу від майстра. Є *M* завдань map і *R* завдань reduce для призначення. *R* визначається конфігурацією, визначеною користувацькою програмою, або конфігурацією за замовчуванням у загальному кластері. Майстер підбирає непрацюючих робітників і присвоює кожному завдання *map*. Після того, як завдання *map* генерують проміжний результат, майстер призначає завдання *reduce* непрацюючим працівникам. Зауважте, що всі завдання *map* повинні бути виконані, перш ніж розпочати будь-яке завдання *reduce*. Це пояснюється тим, що завдання *reduce* беруть на вхід результати з будь-якої задачі *map*, яка може генерувати результат, який потрібно буде консолідувати.
3. Працівник, якому призначено завдання *map*, читає вміст відповідного вхідного розбиття. Він аналізує пари ключів/значень із вхідних даних і передає кожну пару екземпляру визначеної користувачем функції *map*.

Проміжні пари ключ/значення, що виробляються функціями *map*, буферуються в пам'яті на відповідних машинах, які їх виконують.

4. Буферизовані пари періодично записуються на локальний диск і розподіляються на *R* областей за допомогою функції розподілу. Фреймворк надає функцію розділення за замовчуванням, але користувачеві дозволяється перевизначити цю функцію для користувацького розділення. Розташування цих буферизованих пар на локальному диску повертається до головного. Потім головний пересилає ці місця для робітників *reduce*.
5. Коли робітник *reduce* повідомляє майстра про ці місця, він використовує віддалені виклики процедури для зчитування буферизованих даних з локальних дисків працівників *map*. Коли працівник *reduce* прочитав усі проміжні дані, він сортує їх за проміжними ключами (k2 у визначенні, наведеному раніше для функції *reduce*), так що всі входження одного і того ж ключа згруповані разом. Сортування потрібне, оскільки зазвичай багато різних ключів позначають одне і те ж завдання *reduce*. Якщо кількість проміжних даних занадто велика, щоб вміститися в пам'яті, використовується зовнішнє сортування. Знову користувачеві дозволено перевизначити стандартне сортування та групування фреймворку.
6. Після завершення всіх завдань *map* та *reduce* майстер пробуджує програму користувача. У цей момент виклик Map/Reduce у програмі користувача повертається до коду користувача.

Отже, паралельна обробка запитів включає наступні етапи:

- розподіл документів по різних машинах кластеру;
- виконання запиту в рамках задачі *map* на кожній окремій машині кластеру;
- консолідацію результатів запитів на окремих машинах в рамках задачі *reduce.*
- видачу результатів користувачу.

2.5 Висновок

У цьому розділі наведено змістовну постановку задачі обробки надвеликих масивів XML-даних. Проведено аналіз алгоритмів індексації документів у форматі XML та засобів масової паралельної обробки документів. Запропоновано модель представлення XML файлів у якості пласкої структури пар «ключ-значення». Виявлено, що таку структуру можна обробляти у паралельному режимі, із застосуванням можливостей масово розподілених обчислень. Наведено алгоритми для обробки надвеликих масивів даних XML із застосуванням запропонованих методів.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальний опис архітектурного рішення

Для реалізації програмного продукту була обрана платформа Elasticsearch. Для того, щоб забезпечити інтеграцію з існуючим програмним рішенням, необхідно зробити наступне:

1) Вивантажити документ з реляційної бази даних;
2) Трансформувати документ у формат, зрозумілий Elasticsearch (JSON);
3) Вставити документ у сховище, використовуючи Elasticsearch API.

Для зручності всі етапи можна виконувати в рамках однієї програми на мові програмування високого рівня, як-от Python або C#.

На початковому етапі доцільно виконувати вивантаження поступово, щоб не перевантажувати сервер, на якому зберігаються дані – до прикладу, вивантажувати документи кожної ночі в неробочий час.

Щойно документ потрапляє в Elasticsearch, система автоматично додає його до індексу і він стає доступним для операцій пошуку. При підключеному інтерфейсі користувача Kibana додавання нового документа до індексу автоматично поновить будь-які налаштовані робочі панелі тощо.

3.1.1 Вивантаження документу з реляційної бази даних

Для вивантаження документу необхідно виконати SQL-запит. Нехай документ (рис. 3.1) знаходиться у таблиці *Transaction*, яка містить власне документ у вигляді XML та метадані (ідентифікатор, дату останньої зміни, тощо). Тоді для вивантаження документа необхідно завантажити дані в проміжний об’єкт, що містить:

- ідентифікатор документа (далі *id)*;
- власне документ (далі *transaction)*.

Після того, як документ у текстовому вигляді збережений у пам’яті, можна здійснити його трансформацію у JSON.

ImportedTxn.txt

```
<AttestLine>0</AttestLine>
<BaseLoanTermsCompoundNegativeAmort>0</BaseLoanTermsCompoundNegativeAmort>
<Borrowers>
  <Borrower xmlns="http://schemas.bankerssystems.com/2004/ExpereTxn">
    <Aliases />
    <AlimonyChildSupportMaintenancePaymentInd>0</AlimonyChildSupportMaintenancePaymentInd>
    <CitizenType>1</CitizenType>
    <CurrentEmployers>
      <CurrentEmployer>
        <Name>Wolters Kluwer Financial Services</Name>
        <AddressStreetAddr1>6815 Saukview Drive North</AddressStreetAddr1>
        <AddressCity>St Cloud</AddressCity>
        <AddressPostalCode>56303</AddressPostalCode>
        <SelfEmployedInd>0</SelfEmployedInd>
        <PhoneNumber>8002742711</PhoneNumber>
        <Title>Software support</Title>
        <YearsEmployed>5.00</YearsEmployed>
        <YearsInProfession>10.00</YearsInProfession>
        <MonthsInProfessionCount>0</MonthsInProfessionCount>
        <GrossMonthlyIncomeAmount>500.00000</GrossMonthlyIncomeAmount>
        <MonthlyMilitaryEntitlementsAmount>900.00000</MonthlyMilitaryEntitlementsAmount>
        <OtherMonthlyIncomeAmount>120.00000</OtherMonthlyIncomeAmount>
        <SpecialBorrowerEmployerRelationshipInd>0</SpecialBorrowerEmployerRelationshipInd>
        <VerificationOfEmploymentInd>0</VerificationOfEmploymentInd>
        <VerbalVerificationInd>0</VerbalVerificationInd>
        <BonusMonthlyIncomeAmount>700.00000</BonusMonthlyIncomeAmount>
        <CommissionMonthlyIncomeAmount>800.00000</CommissionMonthlyIncomeAmount>
        <OvertimeMonthlyIncomeAmount>600.00000</OvertimeMonthlyIncomeAmount>
        <AddressState>MN</AddressState>
```

Рисунок 3.1 – Приклад документа, що треба обробити

3.1.2 Трансформація XML у JSON

Для трансформації XML у JSON необхідно створити трансформатор, який би підтримував наступні можливості:

- трансформацію XML елемента в JSON елемент;
- запис XPath оригінального XML елемента у новостворений JSON;
- запис додаткових метаданих XML файла із бази даних у JSON.

При трансформації також необхідно зберегти структуру XML документа та інформацію про типи даних з XML: рядки, числа, дати, булеві значення, масиви. Кожна з перелічених вище вимог до трансформації покликана спростити пошук та агрегацію документів користувачами [22].

Для трансформації можна обирати будь-який інструмент, що може створювати JSON. Для середовища .NET найпопулярнішим із таких інструментів є модуль Newtonsoft.JSON. У результаті буде створено JSON-представлення документа (рис. 3.2)

Рисунок 3.2 – JSON-представлення XML-документа.

3.1.3 Вставлення даних в Elasticsearch

Щоб вставити дані в Elasticsearch, необхідно надіслати POST-запит на Elasticsearch API із новоствореним JSON у тілі запиту. Для цього необхідно спершу налаштувати Elasticsearch (рис. 3.3).

Elasticsearch автоматично створить та проіндексує документ у внутрішній базі даних, і документ буде готовий для пошуку, в тому числі з Kibana.

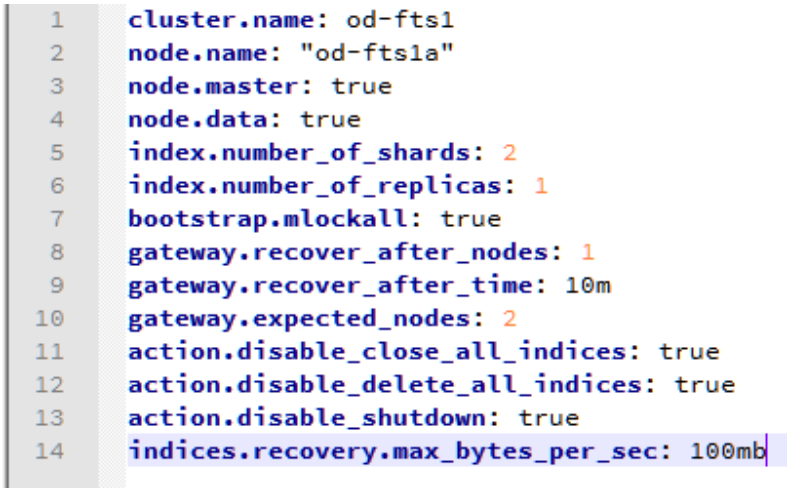


Рисунок 3.3 – Файл конфігурації Elasticsearch

3.2 Опис компонентів програмного забезпечення

3.2.1 Платформа ElasticSearch

Elasticsearch - це система розподіленого пошуку та аналітики в основі Elastic Stack. Logstash та Beats полегшують збір, агрегацію та збагачення ваших даних та зберігання їх у Elasticsearch. Kibana дозволяє вам інтерактивно досліджувати, візуалізувати та ділитися думками щодо своїх даних, а також керувати та контролювати стек. Elasticsearch забезпечує індексацію, пошук та аналіз.

Elasticsearch забезпечує пошук у реальному часі та аналітику для всіх типів даних. Незалежно від того, чи маєте ви структурований чи неструктурований текст, числові дані чи геопросторові дані, Elasticsearch може ефективно зберігати та індексувати його таким чином, що підтримує швидкий пошук. Ви можете вийти за рамки простого пошуку даних та сукупної інформації, щоб виявити тенденції та закономірності ваших даних. Оскільки ваші дані та запити зростають, розподілений характер Elasticsearch дозволяє вашій інфраструктурі безперервно рости разом із ними. [24]

Elasticsearch - це пошукова система з відкритим кодом, побудована на основі Apache Lucene, повнотекстової бібліотеки пошукових систем. Lucene - це, мабуть,

найсучасніша, високопродуктивна та найпопулярніша бібліотека пошукових систем, що існує сьогодні – як серед бібліотек з відкритим кодом, так і пропрієтарних. Але Lucene – це лише бібліотека. Щоб скористатися його потужністю, вам потрібно працювати на Java та інтегрувати Lucene безпосередньо у свою програму. Найгірше, що вам, швидше за все, знадобиться науковий ступінь у галузі пошуку інформації, щоб зрозуміти, як це працює. Люцен дуже складний. Elasticsearch також написаний на Java і використовує Lucene внутрішньо для всієї його індексації та пошуку, але має на меті полегшити повнотекстовий пошук, приховавши складності Lucene за простим, узгодженим API RESTful. Однак Elasticsearch - це набагато більше, ніж просто лучен, і набагато більше, ніж «просто» повнотекстовий пошук. Його можна описати так:

- Розподілене сховище документів, що працює в режимі реального часу, де кожне поле є проіндексованим;
- Розподілений пошуковий движок з аналітикою;
- Можливість масштабування на сотнях серверів для обробки петабайтів даних. [24]

3.2.2 Інструмент Kibana

Kibana - платформа для аналітики та візуалізації з відкритим кодом, розроблена для роботи з Elasticsearch. Ви використовуєте Kibana для пошуку, перегляду та взаємодії з даними, що зберігаються в індексах Elasticsearch. Ви можете легко виконати розширений аналіз даних та візуалізувати свої дані в різних діаграмах, таблицях та картах.

Kibana полегшує розуміння великих обсягів даних. Його простий браузерний інтерфейс дозволяє швидко створювати та ділитися динамічними інформаційними панелями, які відображають зміни в запитах Elasticsearch в режимі реального часу. [25]

Kibana надає користувацький інтерфейс (рис. 2.1) для пошуку серед даних, що зберігаються в Elasticsearch.

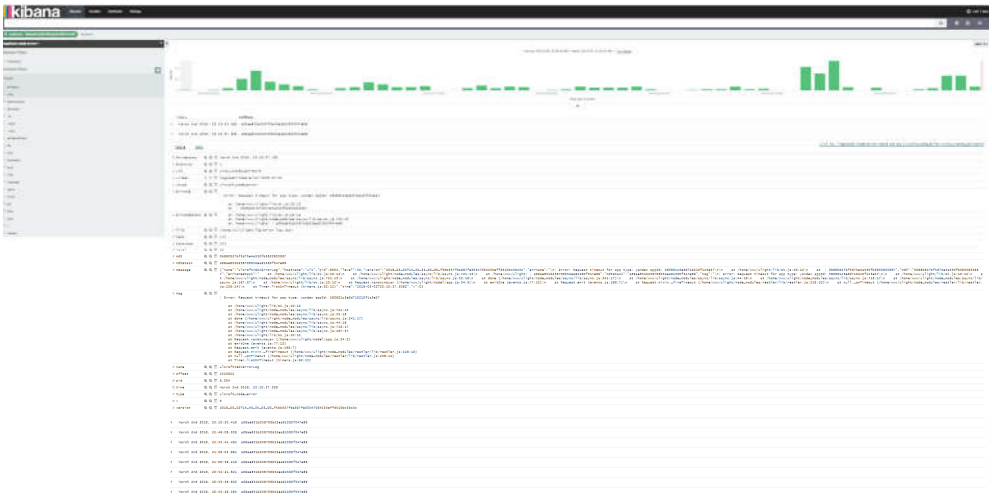


Рисунок 3.4 – Інтерфейс системи пошуку Kibana

Окрім можливостей доступу до даних напряму, Kibana дозволяє користувачам налаштовувати Elascticsearch, створювати робочі панелі, налаштовувати графіки та десятки різноманітних видів візуалізації (рис. 2.2). Система може виводити результати обробки даних в реальному часі, стаючи потужним інструментом для бізнес-аналізу.

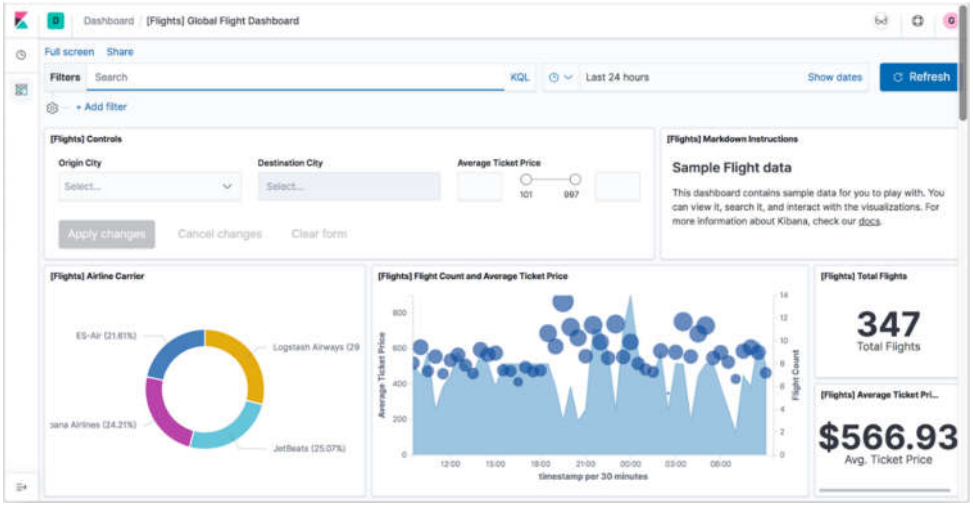


Рисунок 3.5 – Панелі візуалізації Kibana

Отже, Kibana – потужний інструмент з відкритим вихідним кодом, зручний для використання як розробниками, так і кінцевими користувачами.

3.3 Функціональна структура програмного забезпечення

Програмний додаток містить головний файл, файл конфігурації, а також підключається до бази даних та Elasticsearch (рис. 3.4).

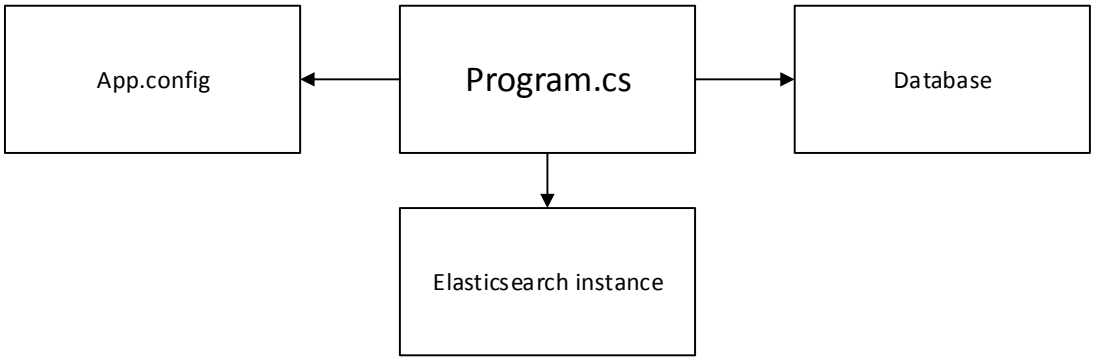


Рисунок 3.4 – Загальна структура ПЗ

3.4 Опис функцій частин програмного забезпечення

На рис. 3.5 зображені основні функції програми.

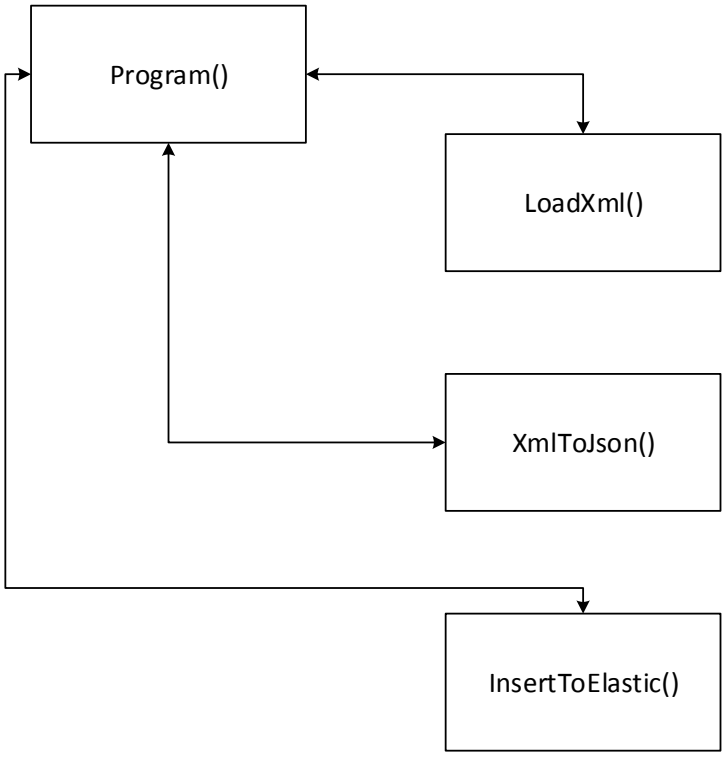


Рисунок 3.6 – Опис функцій користувача

Опис функцій програмного рішення міститься у таблиці 4.1

Таблиця 4.1 – Функції програмного рішення

№ п/п	Назва функції	Вхідні параметри	Вихідні параметри	Призначення
1	main	-	-	Функція запуску програмного рішення та завантаження конфігурації
2	LoadXml	-	XML документ	Функція завантажує XML документ з реляційної бази даних.
3	XmlToJson	XML документ	JSON документ	Функція виконує пошук конвертацію XML документа в JSON формат
4	InsertToElastic	JSON документ		Вставляє зконвертований документ в Elasticsearch

3.5 Інструкція для запуску

Для запуску необхідно встановити та сконфігурувати усе необхідне програмне забезпечення – Microsoft SQL Server, .NET Core Framework, ElasticSearch, Kibana.

Після інсталяції можна запустити програму із командного рядка, вказавши у конфігурації програми необхідні параметри SQL Server та Elasticsearch (рис. 3.6).

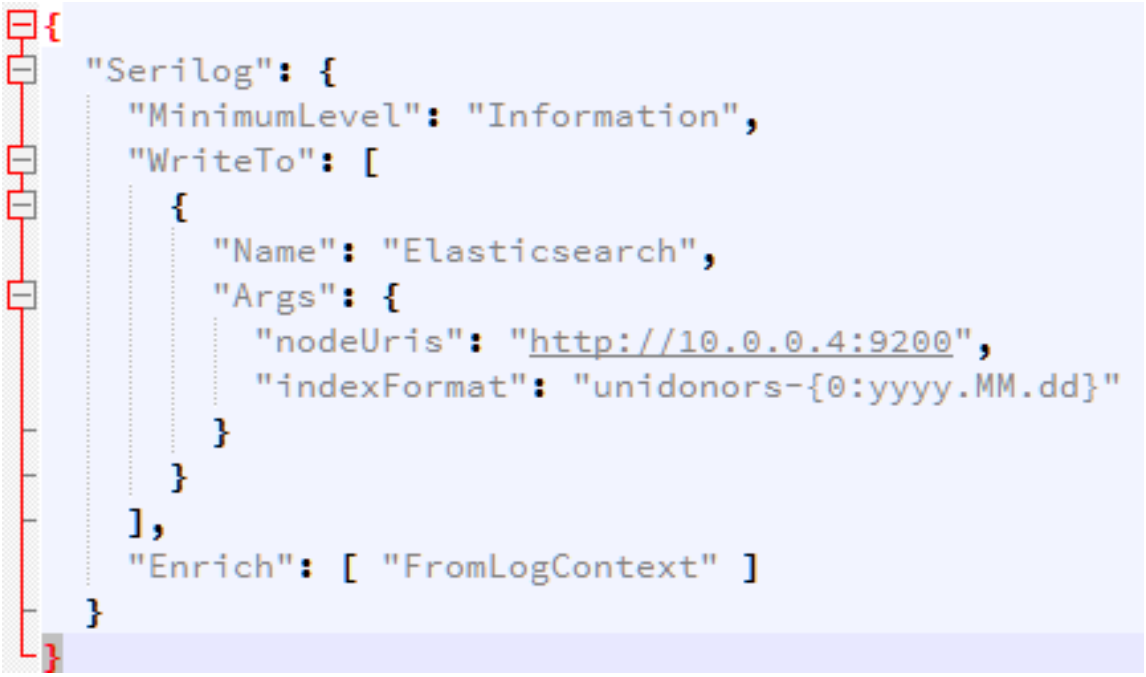


Рисунок 3.7 – Приклад налаштування параметрів програми

Для запуску треба використовувати програму *dotnet run* (рис. 3.7)

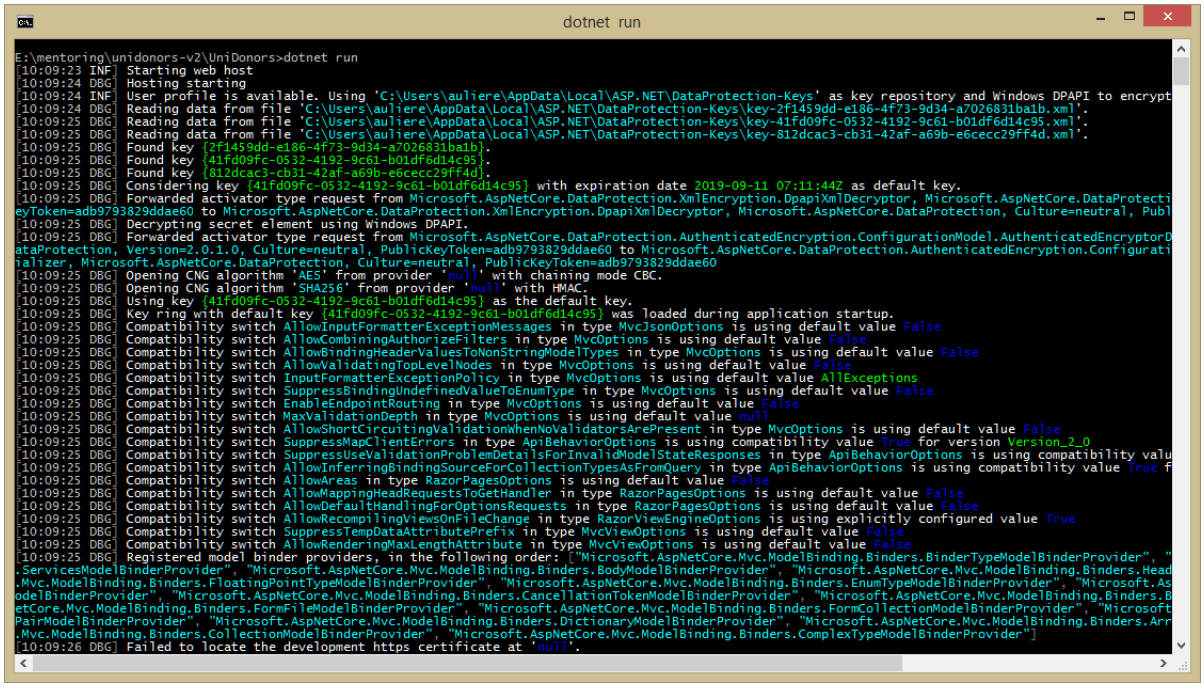


Рисунок 3.8 – Початок роботи проекту

Щоб отрмати результати роботи програми, необхідно зайти на сервер Kibana – за замовчуванням, *localhost:5601* (рис. 3.9-3.15).

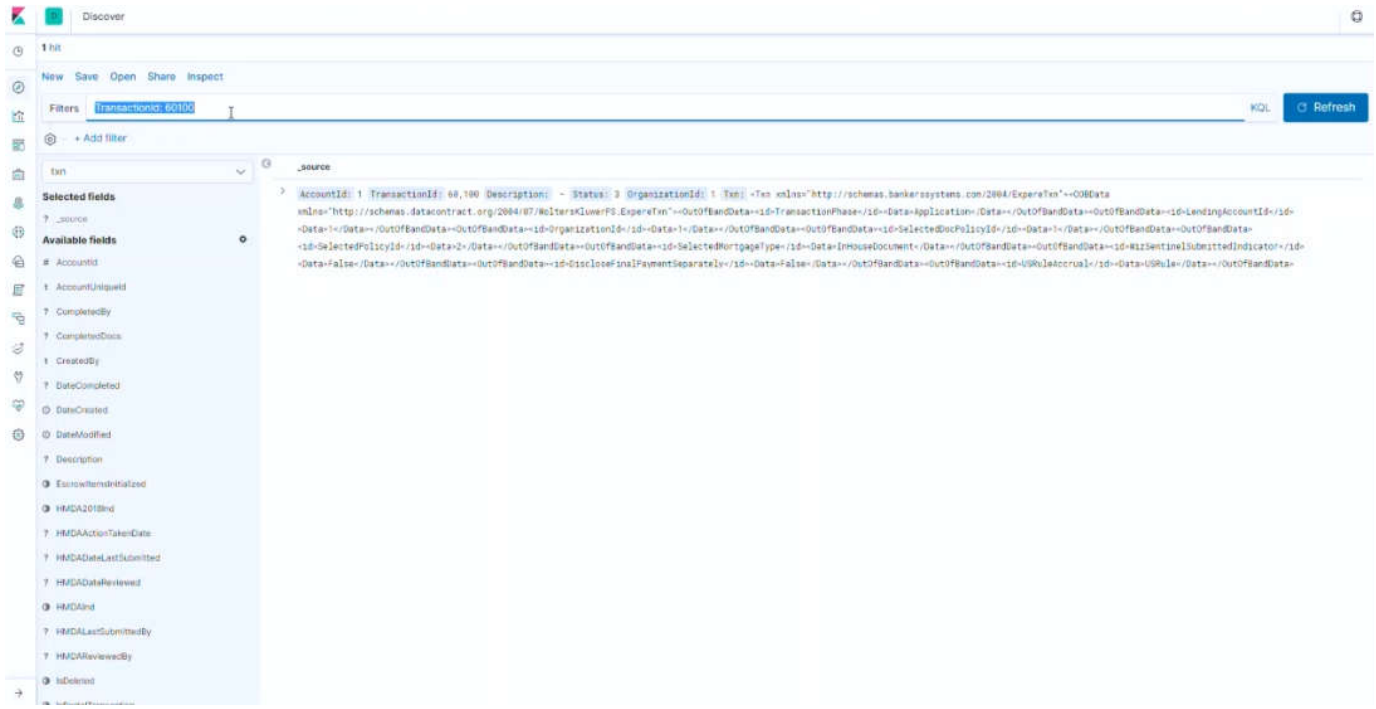


Рисунок 3.9 – Результати роботи. Пошук даних

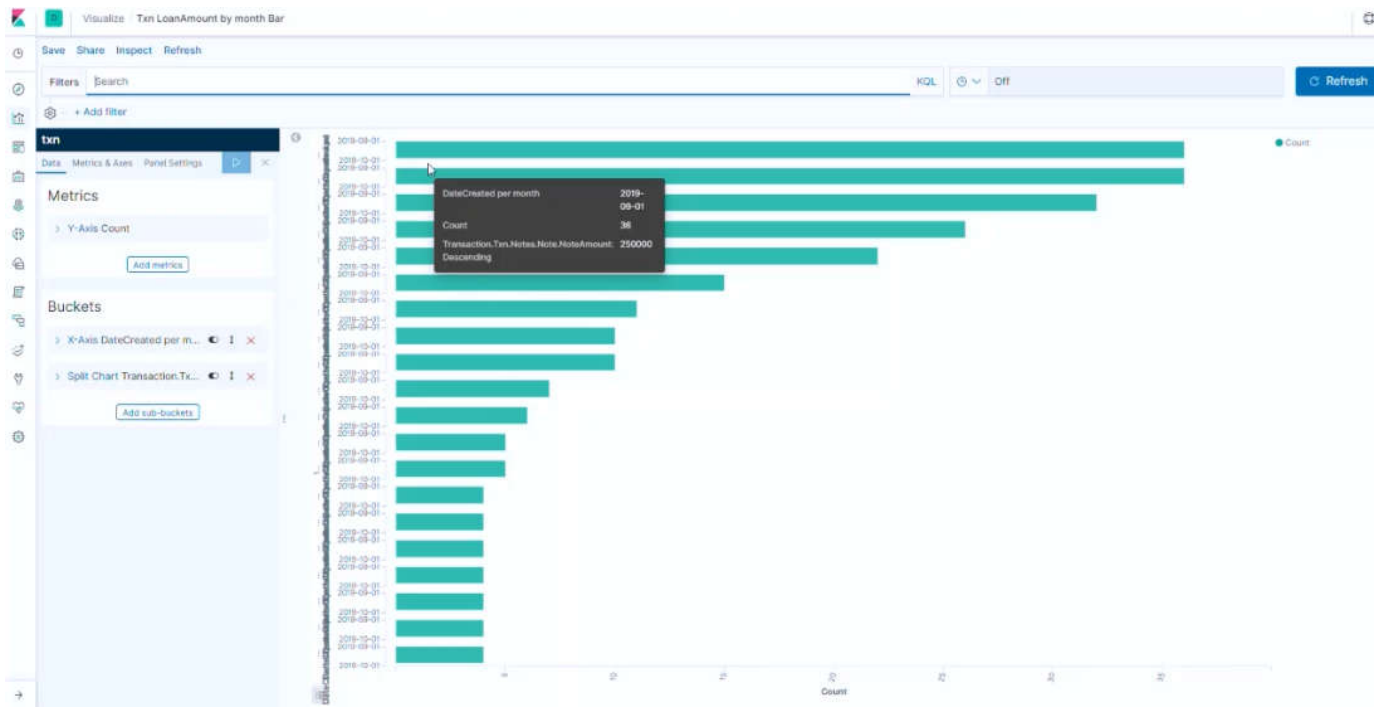


Рисунок 3.10 – Результати роботи. Графік кількості транзакції за проміжок часу

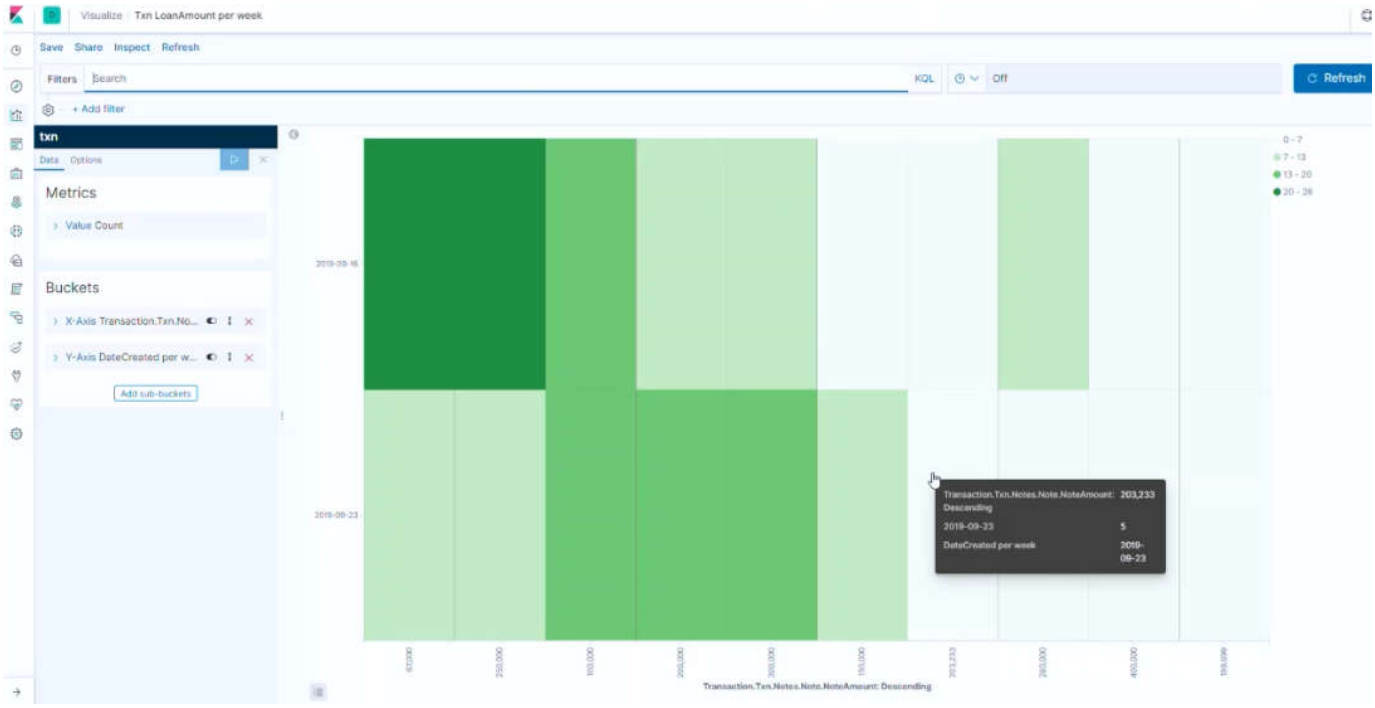


Рисунок 3.11 – Варіант візуалізації даних за допомогою HeatMap

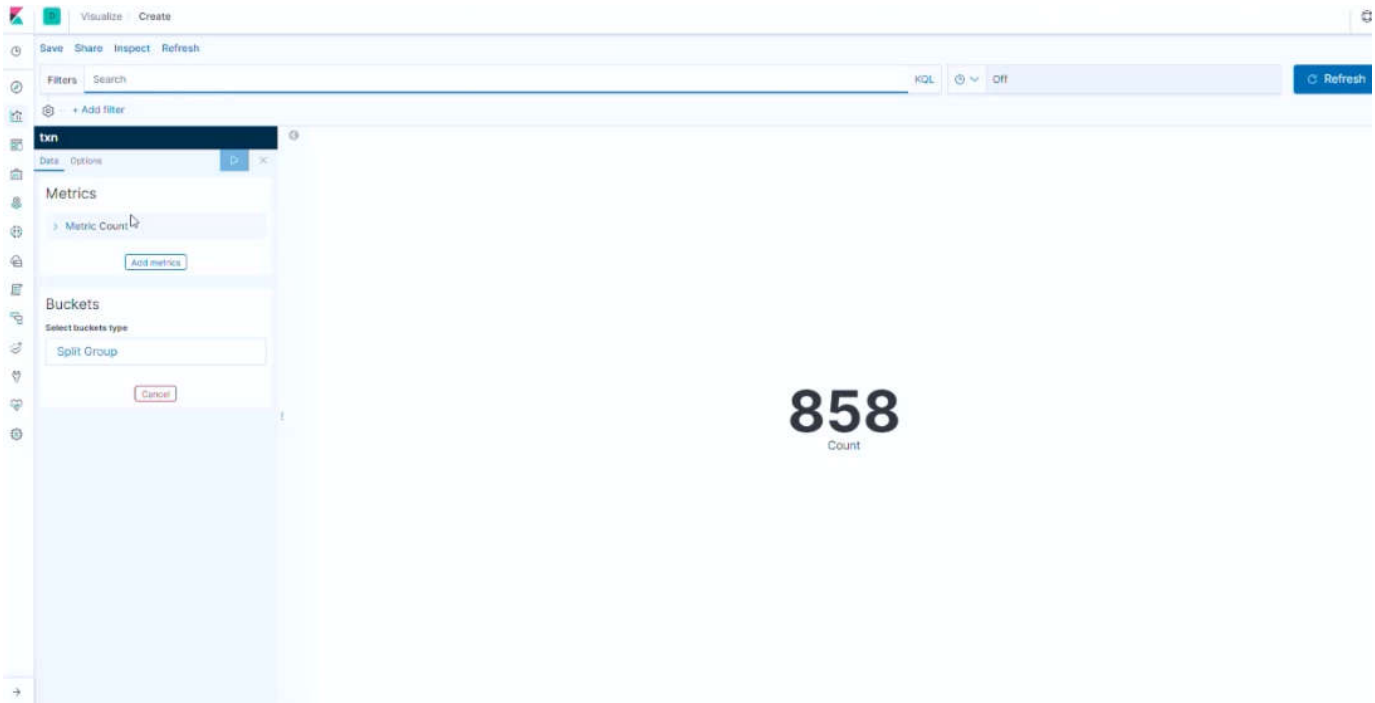


Рисунок 3.12 – Виконання агрегаційного запиту за допомогою інтерфейсу користувача

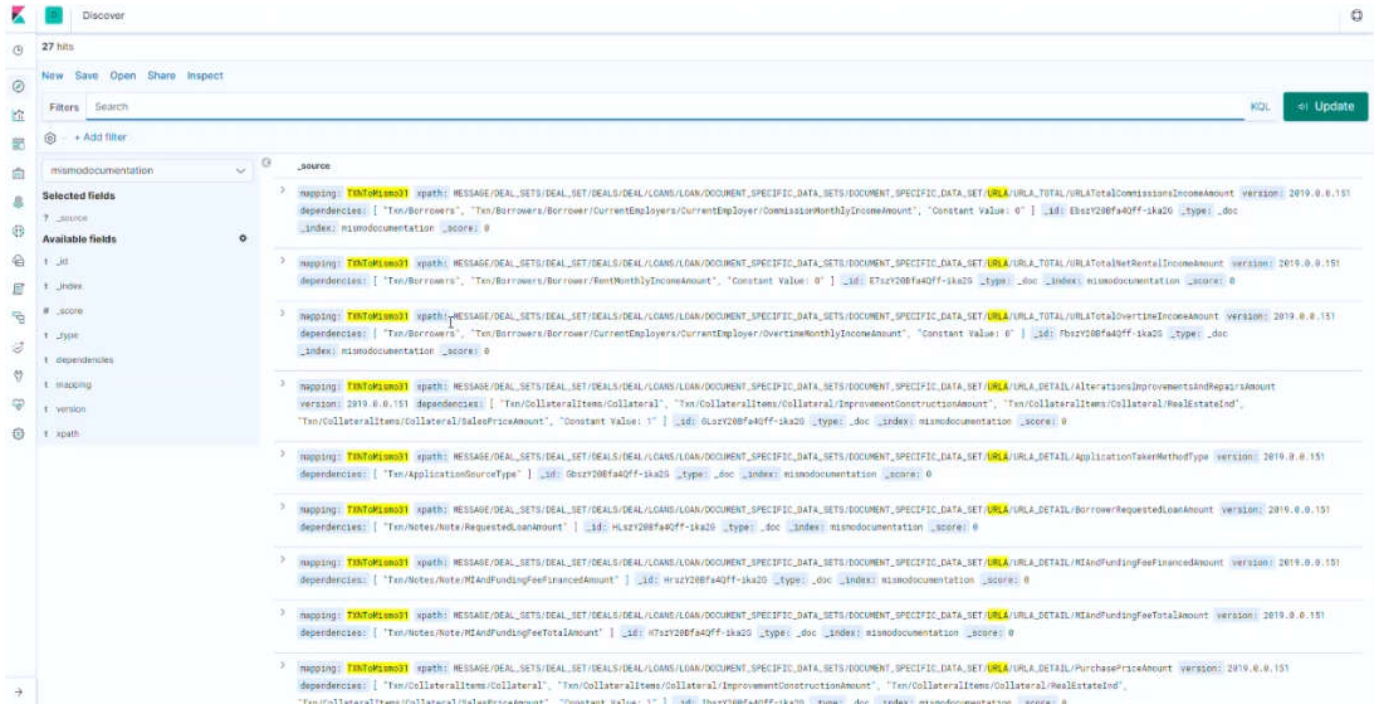


Рисунок 3.13 – Перегляд та фільтрація даних за певним критерієм

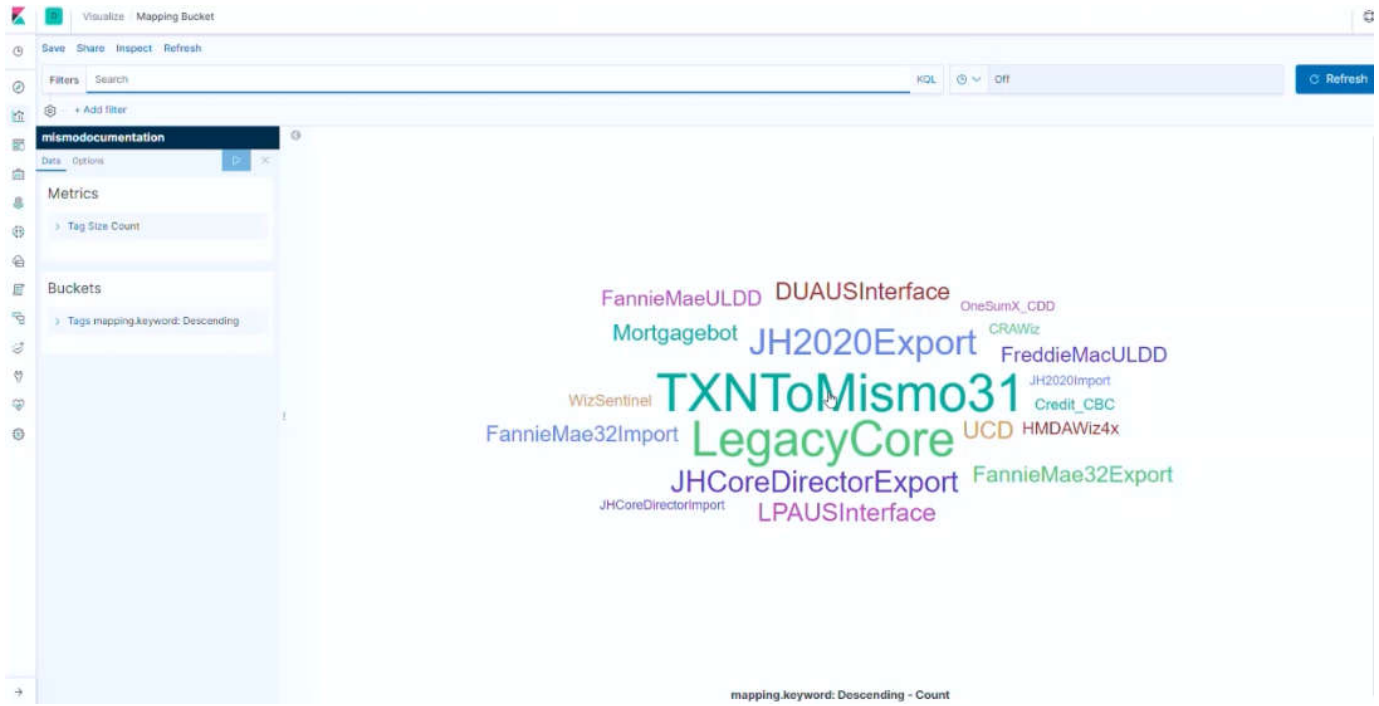


Рисунок 3.14 – Приклад візуалізації даних за допомогою Tag Cloud

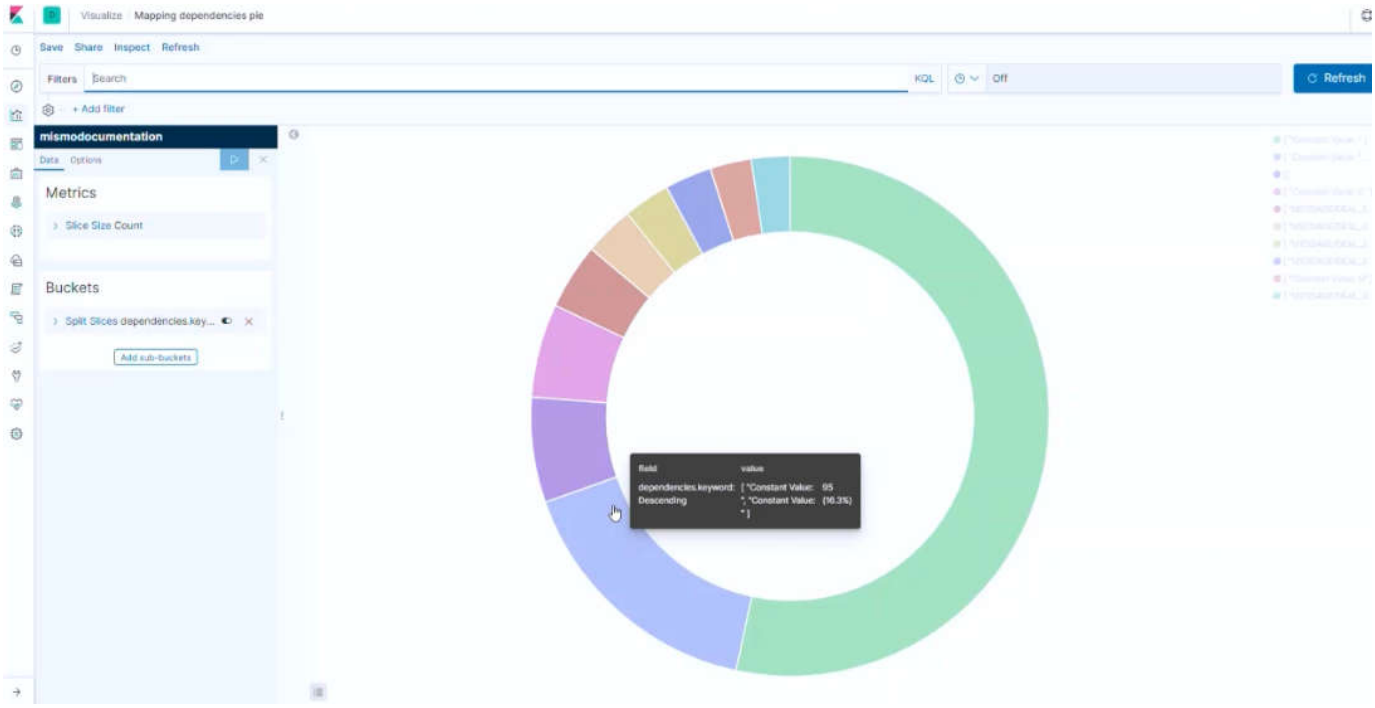


Рисунок 3.15 – Приклад візуалізації даних за допомогою Pie Chart

3.6 Висновок

Була розроблена архітектура програмного забезпечення, та виконана програмна реалізація методу обробки надвеликих масивів даних у форматі XML за допомогою високорівневої мови програмування .NET, а також компонентів ElasticSearch та Kibana. Розгортання платформи організовано за допомогою Docker. Такий вибір дозволяє швидко створити та розгорнути програмне забезпечення, готове для демонстрації та використання кінцевими користувачами.

У цьому розділі описано архітектуру програмного забезпечення, наведено діаграму розгортання програмного забезпечення

4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Опис проблеми та ідеї проекту

Спрямуванням стартап-проекту є розробка платформи, що дозволить користувачам легко розв’язувати задачу аналізу надвеликих масивів XML-даних. В основу платформи покладено розроблені методи ефективної обробки таких даних. У таблиці 4.1 наведено загальний опис проекту.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Аналітична платформа для надвеликих масивів XML-даних	Міграція XML даних	Висока швидкість обробки масивів даних
	Візуальний аналіз XML даних	Отримання онлайн-аналітики для бізнесу та аналітиків
	Аналітичні запити до XML даних	Швидкий доступ до аналітичних даних для розробників

Результат проведеного аналізу ідеї проекту наведено у таблиці 4.2

таблиці 4.2 – Визначення слабких, нейтральних, сильних характеристик ідеї

№ п/п	Техніко-економічні характе-ристики ідеї	(потенційні) товари/концепції конкуренті			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		MS SQL Server	Microsoft PowerBI	JasperSoft BI			
1	Міграція XML даних	+	-	-			X
2	Аналітичні запити до XML даних	-	-	+			X

3	Візуалізація результатів запитів	-	-	+			X
4	Дружність до нетехнічних користувачів	-	+	-		X	
5	Робота з іншими форматами даних	+	+	+	X		

4.2 Технологічний аудит ідеї проекту

Реалізацію ідеї проекту планується проводити на основі існуючих технологій, що доступні у вигляді відкритого програмного забезпечення та підтримуються великими гравцями на ринку обробки даних. Разом з тим, використання розроблених власних алгоритмів дозволить отримати конкурентну перевагу. У таблиці 4.3 розглянуто різні можливі технології реалізації проекту.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Міграція, зберігання XML даних	Oracle XML DB	+	-
		Elasticsearch	+	+
2	Аналітичні запити до XML даних	Oracle XML DB	+	-
		Elasticsearch	+	+
3	Візуалізація результатів запитів	Oracle XML DB	-	-
		Kibana	+	+
Обрана технологія реалізації ідеї проекту: Elasticsearch/Kibana				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення конкуретного середовища та можливостей, які можна використати під час впровадження проекту, а також ринкових ризиків та загроз, що можуть завадити реалізації проекту, дозволить перебачити і спланувати можливі точки зрозстання проекту. Врахування стану ринкового середовища дозволить визначити потенційні проблеми для розвитку проекту. У таблиці 4.4 наведено стан ринку. У таблиці 4.5 наведено опис можливих клієнтів проекту. У таблиці 4.6 наведено аналіз можливих загроз для реалізації проекту зі сторони ринку, а у таблиці 4.7 узагальнено можливості, які сприятимуть швидкому зростанню компанії.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	3 500 000 000
3	Динаміка ринку (якісна оцінка)	зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Uptime сервісу 99.9%
5	Специфічні вимоги до стандартизації та сертифікації	немає
6	Середня норма рентабельності в галузі (або по ринку), %	150%

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба у аналізі надвеликих масивів XML даних	Бізнес, що зберігає дані у форматі XML (розробники банківськихі медичних систем тощо)	Великі компанії можуть спробувати створити свої засоби розробки.	До продукції: -точність обробки даних -швидкодія -можливість встановлення на власних серверах
				До постачальника: - технічна підтримка користувачів - впровадження нових можливостей у систему - навчання користувачів системи

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Репутація	Клієнти не ризикують отримувати послуги від вендора з невідомою репутацією	Просування бренду компанії; Участь у тематичних виставках та конференціях; Безкоштовні пробні версії продукту.
2	Впровадження	Небажання клієнта впроваджувати нове ПЗ	Самостійне переведення клієнта на свою продукцію, їх навчання і техпідтримка

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Популярність XML формату	Стандарти багатьох галузей передбачають використання саме XML формату для обміну даними	Створення продукту для інтеграції з сервісами третіх осіб для збору даних
2	Machine Learning	Компанії хочуть застосувати Machine Learning для виявлення закономірностей у своїх даних	Запровадження підходів Machine Learning у продукті для подальшої автоматизації аналізу даних

4.4 Аналіз конкуренції на ринку

У таблицях 4.8 та 4.9 наведено загальні риси конкуренції на ринку.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Чистий тип конкуренції	Наявність великої кількості конкурентів	Ускладнене виявлення конкуретних переваг
2. Міжнародний рівень конкурентної боротьби	Розміщення проектів-конкурентів	Підтримка англійської мови та мов локальних користувачів. Підтримка актуальної документації мовами користувачів. Можливість подорожувати для підтримки та запровадження продукту.
3. Внутрішньогалузева конкуренція	Визначена спрямованість (інформаційні технології)	Існують інструменти і методи розробки, тестуваннння, впровадження

4. Конкуренція за видами товарів - між бажаннями	Конкуренцію виграє той проект, що більше задовольняє клієнтів	Користувачеві потрібно донести конкурентні переваги
5. За характером конкурентних переваг - цінова	Користувач імовірно обере дешевший проект	Ціна має бути ринковою
6. За інтенсивністю - марочна	Унікальне ім’я	Можливість досягти впізнаваності проекту шляхом просування

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Oracle XML DB IBM DB2		Oracle IBM	Бізнес, що зберігає дані у форматі XML	MongoDB
Висновки	Середня інтенсивність конкурентної бортьби	Є можливості входу в ринок, які не використовуються	Постачальники не диктують умови роботи на ринку.	Клієнти диктують умови роботи на ринку. Продукт повинен бути швидким, зручним та приємним оку щоб затримати увагу користувача на собі	Необхідно забезпечувати конкурентоспроможну швидкодію.

Отже, з огляду на конкурентну ситуацію на ринку рішень для обробки надвеликих масивів XML-даних існує принципова можливість виходу на цей ринок. Аби наш проект був конкуретноспроможним, необхідно забезпечити швидкодію продукту, легкість та зручність використання для кінцевих користувачів. Вагомою перевагою також буде можливість аналітичної обробки XML даних.

4.5 Стратегія розвитку продукту

Результати аналізу факторів конкурентоспроможності наведено у таблиці 4.10. За визначеними факторами конкурентоспроможності проаналізовано сильні та слабкі сторони стартап-проекту у таблиці 4.11. SWOT-аналіз стартап-проекту (таблиця 4.12) є основою для розробки альтернативних стратегії впровадження на ринку (таблиця 4.13). У таблицях 4.14-4.18 описано ключові позиції стратегії розвитку продукту.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Швидкість оброблення даних	Швидкість оброблення даних вища, ніж у конкурентів
2	Аналітична обробка XML	Лише один з конкурентів реалізував дану функцію
3	Візуалізація результатів обробки	Сучасний вигляд візуалізації

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін «OLAP4XML»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з Oracle XML DB						
			-3	-2	-1	0	+1	+2	+3
1	Швидкість оброблення даних	16				X		X	
2	Аналітична обробка XML	20							X
3	Візуалізація результатів обробки	18						X	

Таблиця 4.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Швидкість оброблення даних Аналітична обробка XML Візуалізація результатів обробки	Слабкі сторони: Можливість оброблення різних форматів даних Технічна підтримка користувачів
Можливості: Популяризація Вбудовані запити Автоматичний аналіз даних на основі XML-схеми	Загрози: Невідомість бренду Консервативність клієнтів

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Створення платформи, що дозволяє завантажити та проаналізувати XML дані.	Середня, за рахунок попиту в аудиторії та малої кількості конкурентів у цьому напрямку	4 місяці
2	Продаж прав на технологію більшій компанії що займається цим же напрямом	Велика, за рахунок потенційного зменшення конкурентів у покупця з запозиченням робочої технології.	1 місяць

Таблиця 4.14 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Створення платформи, що дозволяє завантажити та проаналізувати XML дані.	Стратегія диферен-ційованого маркетингу	Швидкість оброблення даних Аналітична обробка XML Візуалізація результатів обробки	Стратегія спеціалізації

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Розробники ПЗ, що зберігають дані у фоматі XML	Так	Є	Середня	Складно
2	Інтегратори що збирають дані у форматі XML	Так	Є	Середня	Середньо
Які цільові групи обрано: 2					

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першо-прохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Шукати нових споживачів	Ні	Стратегія заняття конкурентної ніші

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Цілодобова доступність	Стратегія спеціалізації	Швидкість оброблення даних	Просто, зручно, точно
	Надійний захист данних		Аналітична обробка XML	
	Аналітика XML даних		Візуалізація результатів обробки	
	Зручні інструменти роботи			
	Швидкодія			

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Аналіз надвеликих масивів XML даних	Візуальний аналіз даних	Можливість швидкої аналітичної обробки XML даних

Таблиця 4.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>Товар за задумом</td><td colspan="3">Спосіб аналізу надвеликих масивів XML даних</td></tr>
<tr><td rowspan="5">Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Вартість</td><td rowspan="3"></td><td rowspan="3"></td></tr>
<tr><td>2. Відповідність моді</td></tr>
<tr><td>3. Стабільність роботи</td></tr>
<tr><td colspan="3">Якість: Велика точність визначення частин обличчя і напряму погляду</td></tr>
<tr><td></td><td colspan="3">Пакування: розповсюдження за моделлю SaaS, On Premise, документація</td></tr>
<tr><td></td><td colspan="3">Марка: OLAP4XML</td></tr>
<tr><td>Товар із підкріпленням</td><td colspan="3">Після продажу</td></tr>
<tr><td colspan="4">Закритий вихідний код, зареєстрована торгова марка, патенг на алгоритм</td></tr>
</table>

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на послугу
1	37742000 грн.	53900 грн	Не важливо	53900-37742000 грн.

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Закупівля товарів та послуг програмного забезпечення	Розгортання програмного забезпечення Тренування персоналу для використання ПЗ	1	Відділ збуту - користувач

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користую-ться цільові клієнти	Ключові позиції, обрані для позиціону-вання	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Збирання великих об’ємів XML даних	Інтернет, конференції	1. Аналіз масивів XML даних 2. Візуалізація аналізу	1. Донести до споживача наявність та назву продукту 2. Донести конкурентні переваги 3. Переконати у необхідності придбання продукту	Демонстрація роботи

4.6 Висновок

У цьому розділі була обґрунтована можливість створення стартап-проекту на основі розробленого методу та програмного забезпечення для обробки надвеликих масивів даних у форматі XML. Було наведено характеристику стартап-проекту, проаналізовано ситуацію на потенційному ринку для запуску проекту, визначено стратегію розвитку підприємства.

5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОБОТИ АЛГОРИТМУ

Для дослідження ефективності роботи алгоритму було проведено ряд досліджень із використанням трьох альтернативних методів обробки надвеликих масивів XML даних. Короткий опис цих методів наведено у таблиці 5.1

Таблиця 5.1 – Методи, що порівнюються

Назва	Характеристика	Коментар
Метод 1 (M1)	Зберігання даних XML у вигляді індексу документів, оброблення запитів у паралельному режимі	Метод, запропонований автором роботи
Метод 2 (M2)	Зберігання даних у реляційній базі даних (MS SQL), використання механізмів SQL для обробки -запитів	Метод описано у роботі [6]
Метод 3 (M3)	Зберігання документів у файловій системі, обробка запитів за допомогою відомого методу обробки запитів XSLT із використанням XQuery.	Метод описано у роботі [15]

5.1 Опис експерименту

Для забезпечення рівних умов для випробування різних методів обробки надвеликих масивів XML даних усі експерименти проводилися у одному середовищі:

- Операційна система: Microsoft Windows 10;
- центральний процесор: AMD A8-4500M @ 2 GHz;
- оперативна пам’ять: 7,2 GB DDR3 @ 1600 MHz;
- постійна пам’ять: жорсткий диск;

Усі методи були випробувані на таких наборах даних: [21]

- Protein Sequence Database: розмір 683 MB, глибина дерева – 7 (H_1).
- NASA: розмір 23 MB, глибина дерева – 8 (H_2).

– Mondial: розмір 1 МВ, глибина дерева – 5 (H_3).

Робота методів обробки надвеликих масивів XML оцінювалася за такими показниками:

– Зайнятий записаними даними обсяг пам'яті на диску (V)

– Час, необхідний для запису даних в систему зберігання (T_{in})

– Час, необхідний для виконання простого запиту (T_1)

– Час, необхідний для виконання аналітичного запиту (T_2)

5.2 Результати експерименту

Експеримент було повторено 5 разів із тими самими вхідними даними, після чого були визначені середні значення вищевказаних показників. У таблицях 5.2-5.6 наведено результати експеритменту.

Таблиця 5.2 – Перший запуск експеритменту

	Метод 1 (М1)			Метод 2 (М2)			Метод 3 (М3)		
	H_1	H_2	H_3	H_1	H_2	H_3	H_1	H_2	H_3
V (Мб)	2392,5	91,8	3,78	1745,1	69,8	2,53	683	23	1
T_{in} (сек)	17,283	5,495	1,493	10,964	2,734	0,546	0,01	0,01	0,01
T_2 (сек)	0,021	0,001	0,0001	392,59	65,54	1,1	849,1	183,58	68,3
T_3 (сек)	0,093	0,002	0,0001	501,34	71,3	1,9	1211	285,7	98,03

Таблиця 5.3 – Другий запуск експеритменту

	Метод 1 (М1)			Метод 2 (М2)			Метод 3 (М3)		
	H_1	H_2	H_3	H_1	H_2	H_3	H_1	H_2	H_3
V (Мб)	2392,5	91,8	3,78	1743,1	68,8	2,73	683	23	1
T_{in} (сек)	19,332	4,235	1,220	10,935	3,719	0,632	0,01	0,01	0,01
T_2 (сек)	0,021	0,001	0,0001	345,34	68,32	0,92	831,6	189,93	69,5
T_3 (сек)	0,083	0,002	0,0001	488,45	73,4	1,95	1218	295,6	101,13

Таблиця 5.4 – Третій запуск експеритменту

	Метод 1 (М1)			Метод 2 (М2)			Метод 3 (М3)		
	H_1	H_2	H_3	H_1	H_2	H_3	H_1	H_2	H_3
V (Мб)	2392,5	91,8	3,78	1743,1	68,8	3,01	683	23	1
T_{in} (сек)	18,611	4,722	1,237	11,010	3,730	0,523	0,01	0,01	0,01
T_2 (сек)	0,021	0,001	0,0001	345,34	65,54	1,05	859,9	186,4	70,4
T_3 (сек)	0,095	0,002	0,0001	488,45	72,1	1,87	1205	283,4	95,49

Таблиця 5.5 – Четвертий запуск експеритменту

	Метод 1 (М1)			Метод 2 (М2)			Метод 3 (М3)		
	H_1	H_2	H_3	H_1	H_2	H_3	H_1	H_2	H_3
V (Мб)	2392,5	91,8	3,78	1743,1	68,8	2,75	683	23	1
T_{in} (сек)	18,710	5,074	1,576	10,412	2,892	0,595	0,01	0,01	0,01
T_2 (сек)	0,021	0,001	0,0001	345,34	65,54	1,15	844,3	179,23	67,4
T_3 (сек)	0,089	0,002	0,0001	488,45	74,8	1,83	1234	245,2	91,3

Таблиця 5.6 – П'ятий запуск експеритменту

	Метод 1 (М1)			Метод 2 (М2)			Метод 3 (М3)		
	H_1	H_2	H_3	H_1	H_2	H_3	H_1	H_2	H_3
V (Мб)	2392,5	91,8	3,78	1743,1	68,8	2,63	683	23	1
T_{in} (сек)	17,574	4,346	1,435	10,368	3,724	0,610	0,01	0,01	0,01
T_2 (сек)	0,021	0,001	0,0001	345,34	65,54	1,09	859,5	183,58	68,7
T_3 (сек)	0,092	0,002	0,0001	488,45	70,54	1,91	1195	281,1	99,5

У таблиці 5.7 наведено усереднені значення показників результатів експерименту, що дозволяє побачити більш реальну оцінку результатів цього експерименту.

Таблиця 5.7 – Середні значення вимірюваних показників

	Метод 1 (М1)			Метод 2 (М2)			Метод 3 (М3)		
	H_1	H_2	H_3	H_1	H_2	H_3	H_1	H_2	H_3
V (Мб)	2392,5	91,8	3,78	1743,5	69,01	2,73	683	23	1
T_{in} (сек)	18,302	4,7744	1,3922	10,738	3,3598	0,5812	0,01	0,01	0,01
T_2 (сек)	0,021	0,001	0,0001	354,79	66,096	1,062	848,88	184,54	68,86
T_3 (сек)	0,0904	0,002	0,0001	491,03	72,428	1,892	1212,6	278,2	97,09

Побудуємо графіки, на яких зобразимо результат експерименту для наочного порівняння (рис. 5.1 – 5.12)

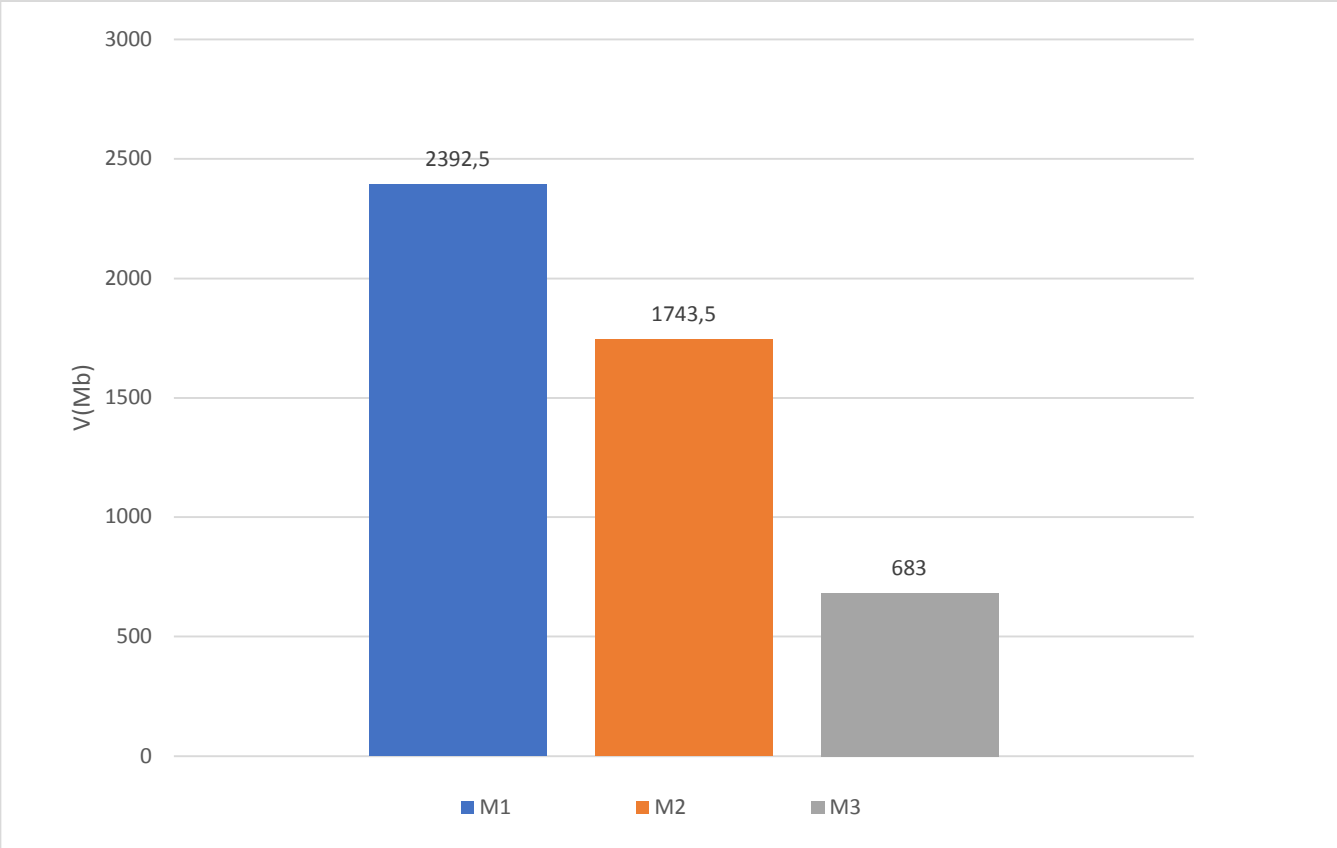


Рисунок 5.1 – Порівняння обсягу даних у сховищі для першого набору даних (H1)

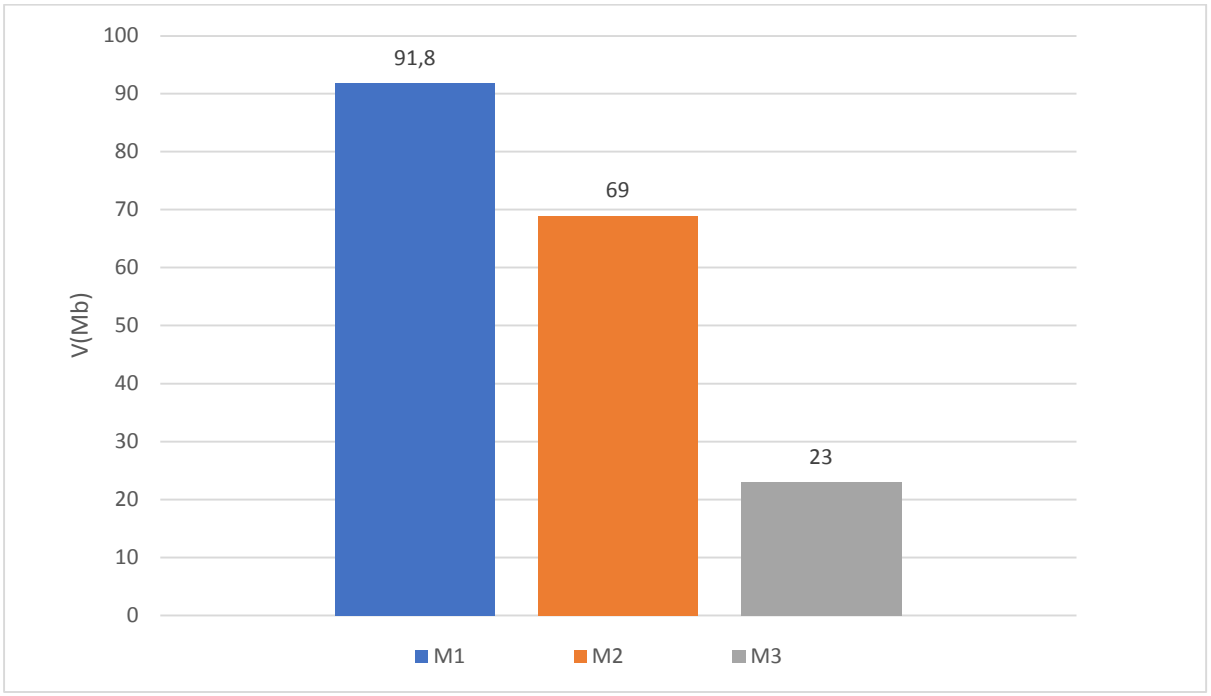


Рисунок 5.2 – Порівняння обсягу даних у сховищі для другого набору даних (H2)

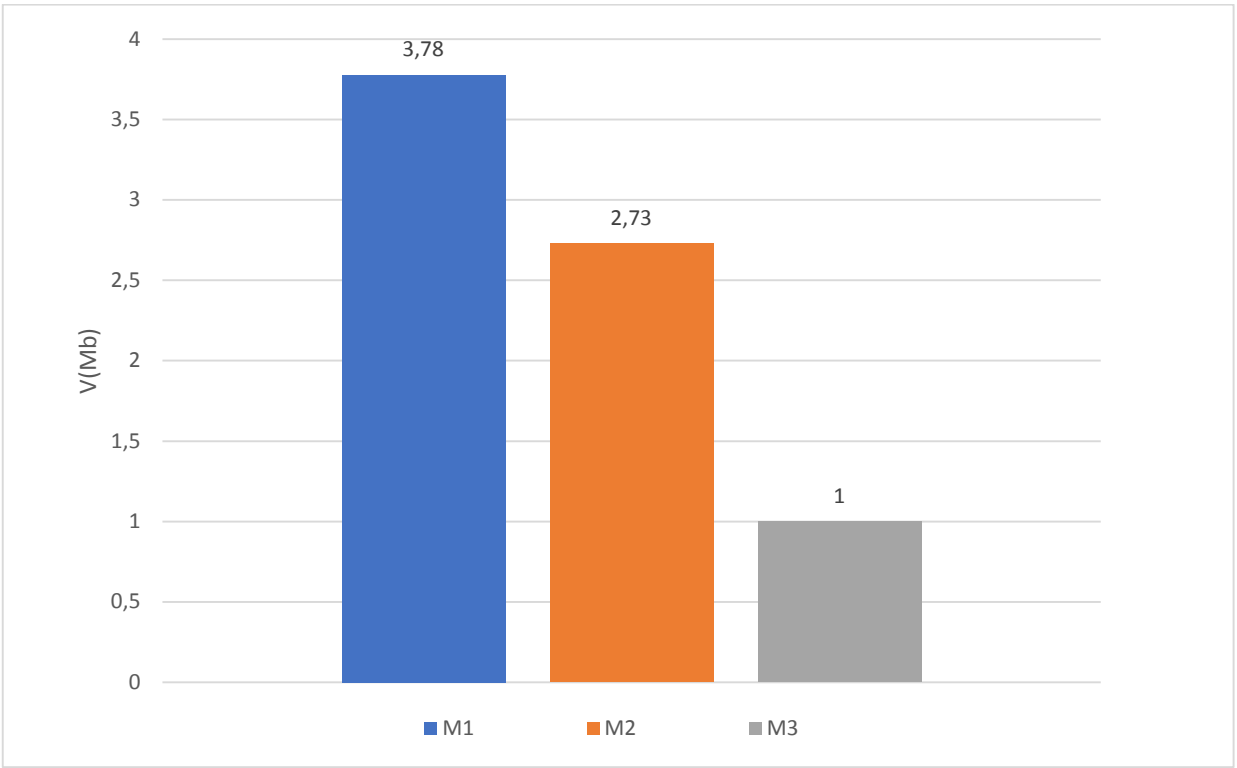


Рисунок 5.3 – Порівняння обсягу даних у сховищі для третього набору даних (H3)

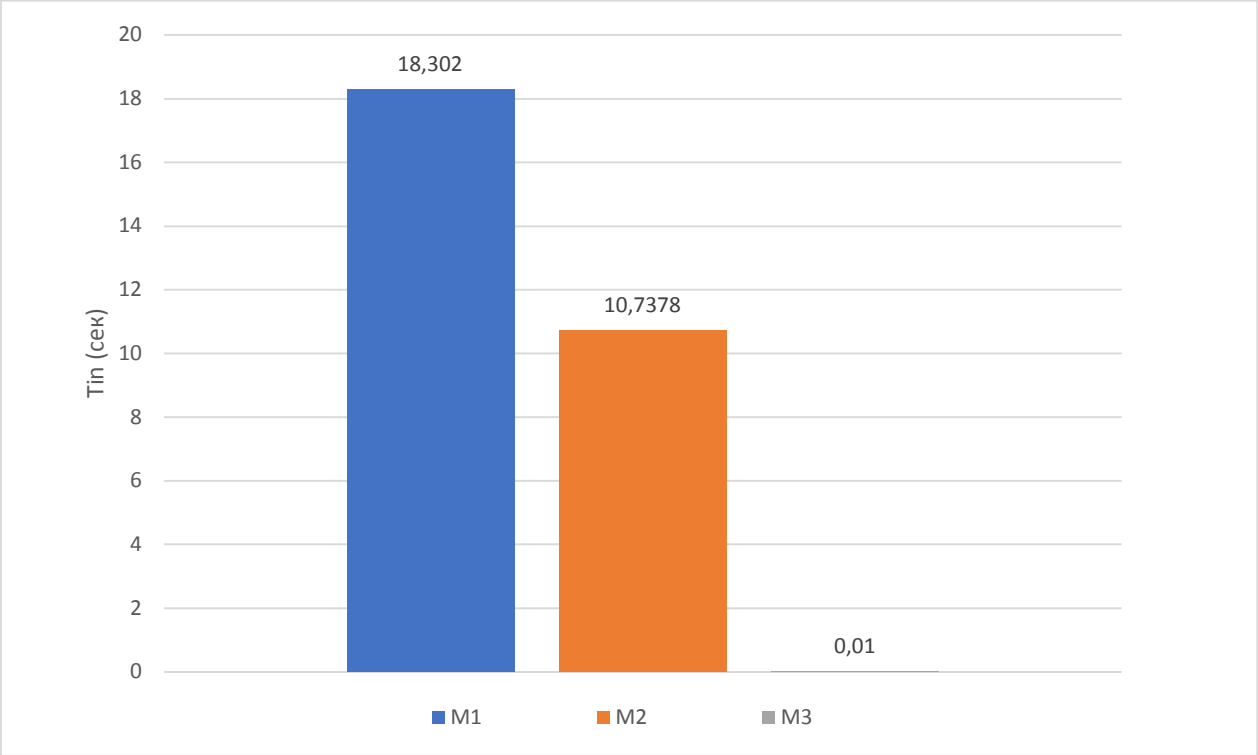


Рисунок 5.4 – Порівняння часу вставки даних у сховище (набір даних H1)

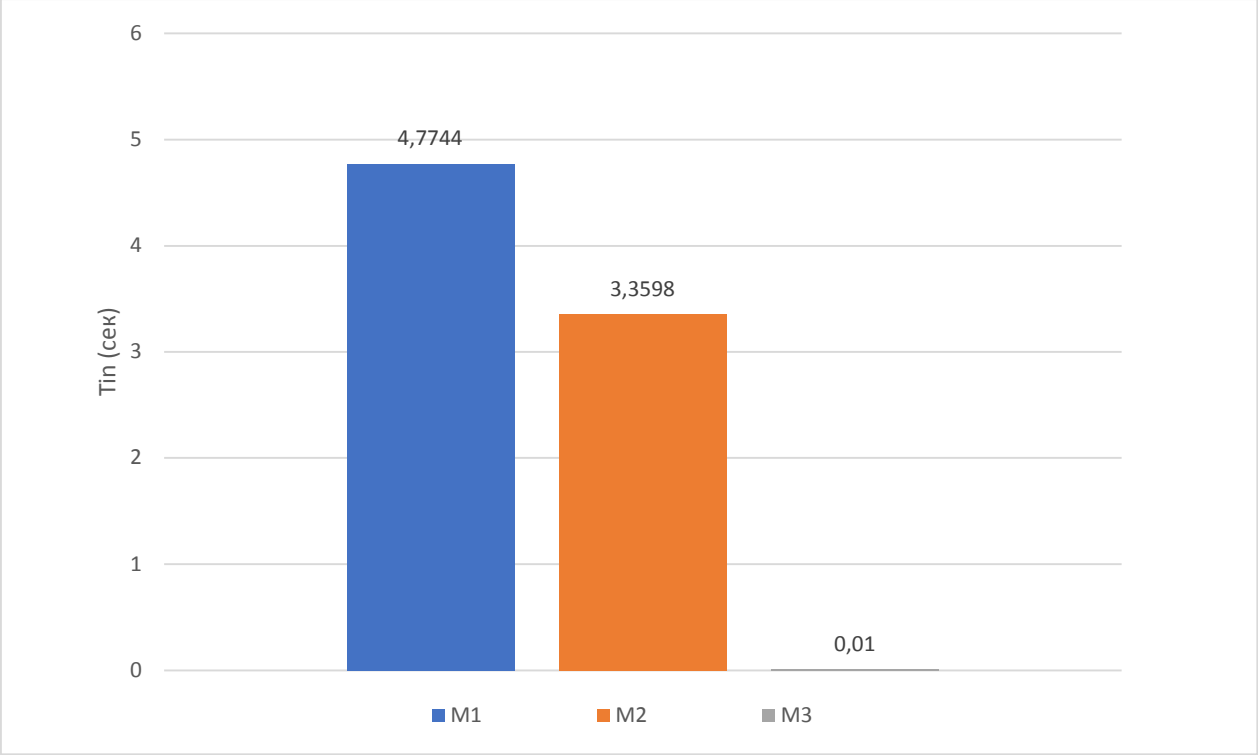


Рисунок 5.5 – Порівняння часу вставки даних у сховище (набір даних H2)

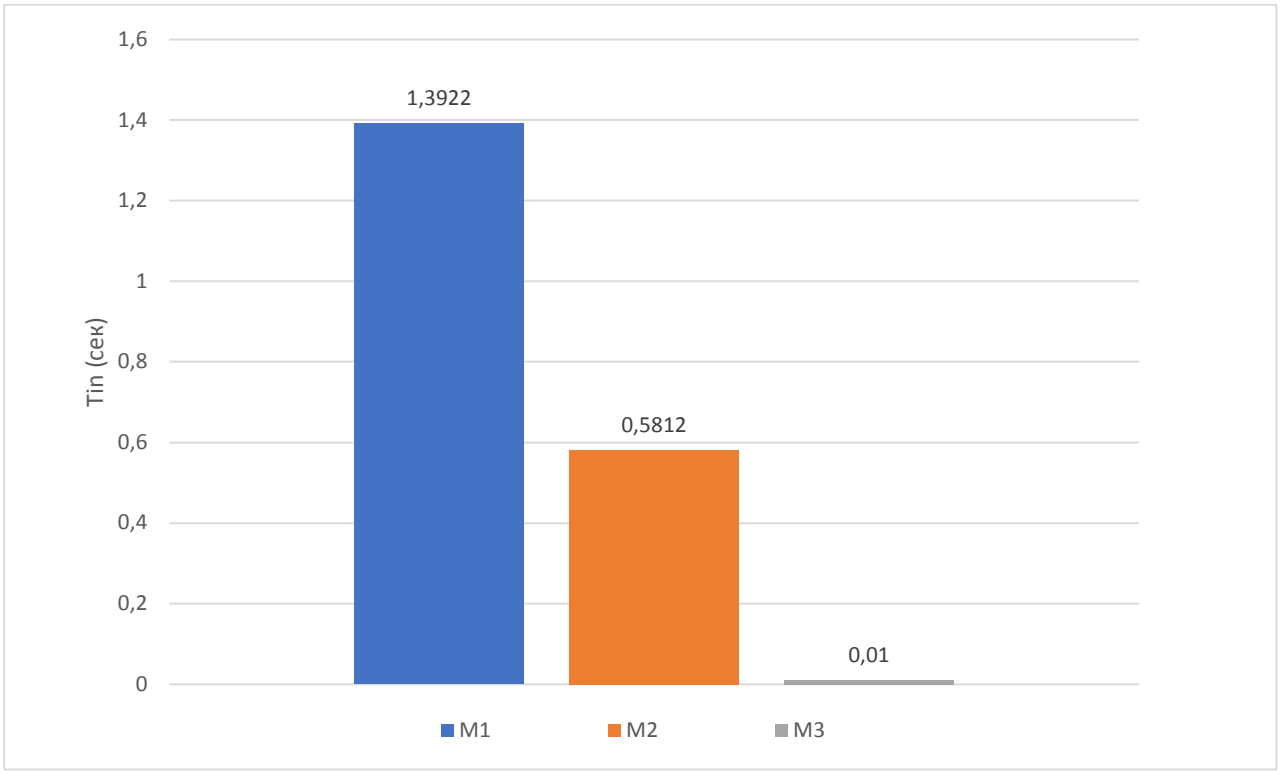


Рисунок 5.6 – Порівняння часу вставки даних у сховище (набір даних H3)

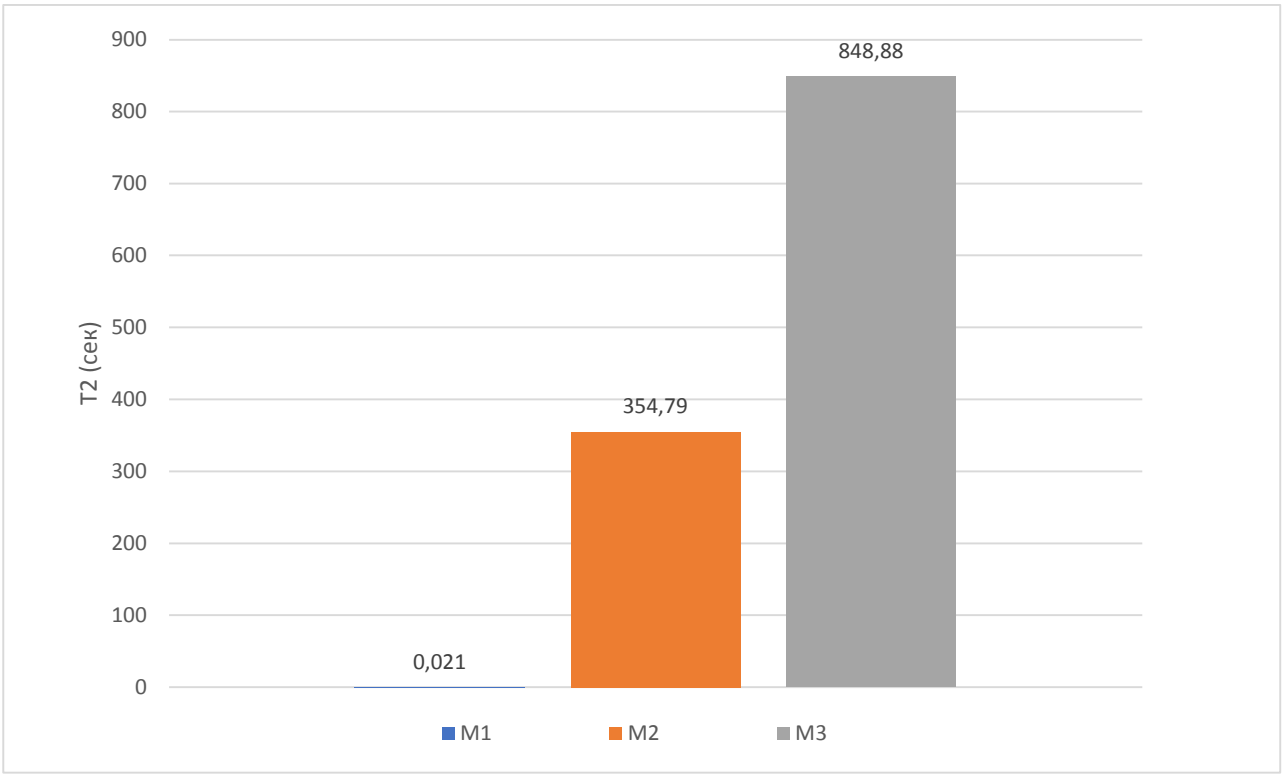


Рисунок 5.7 – Порівняння часу простої вибірки зі сховища (набір даних H1)

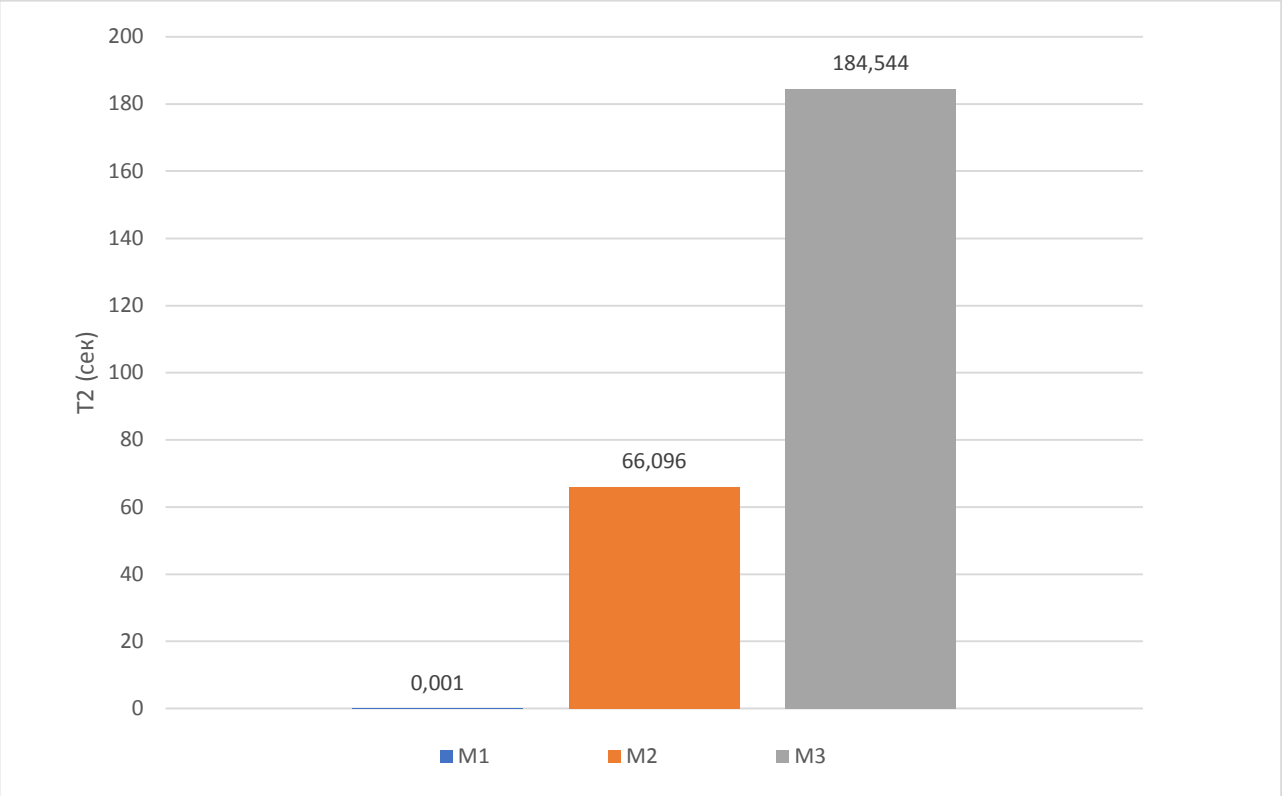


Рисунок 5.8 – Порівняння часу простої вибірки зі сховища (набір даних H2)

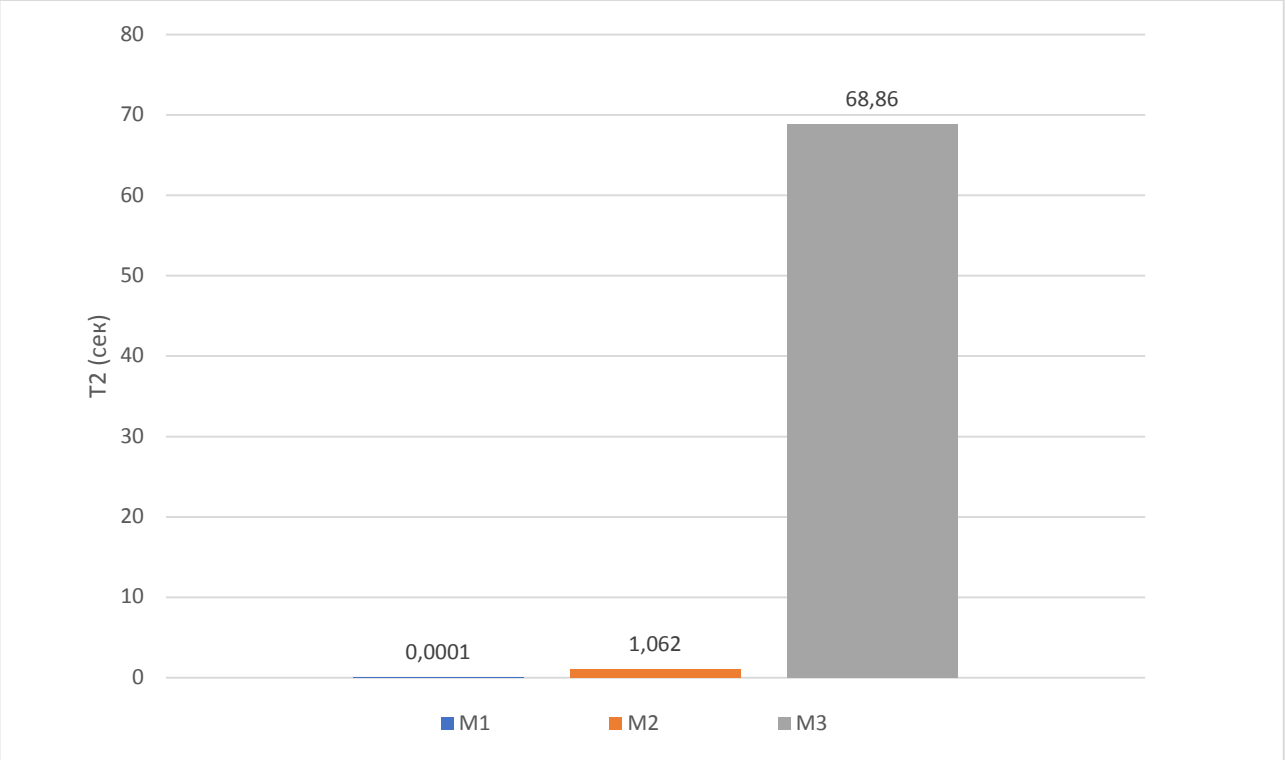


Рисунок 5.9 – Порівняння часу простої вибірки зі сховища (набір даних H3)

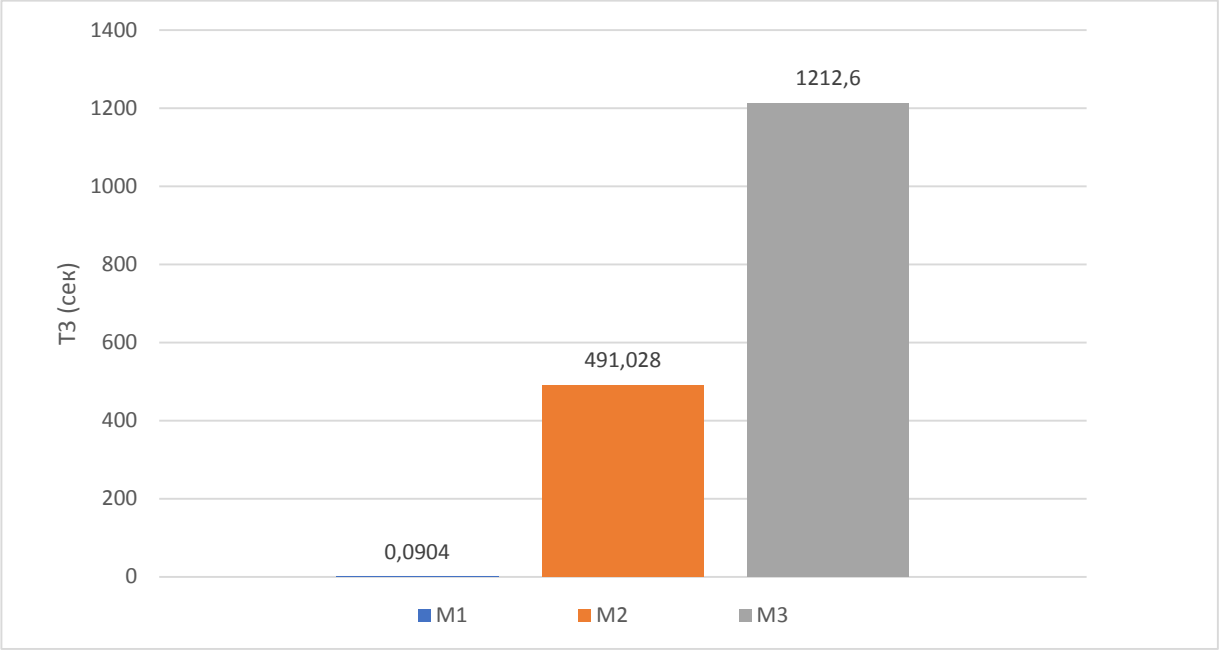


Рисунок 5.10 – Порівняння часу агрегаційної вибірки зі сховища (набір даних H1)

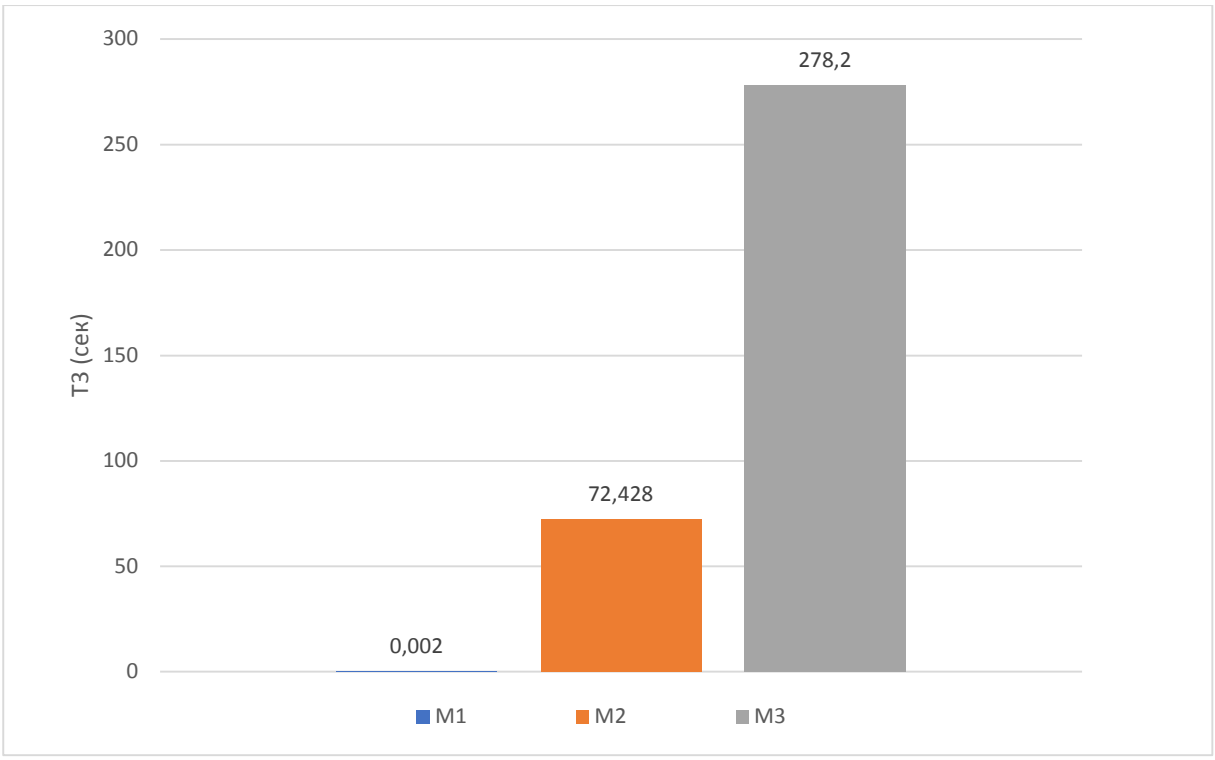


Рисунок 5.11 – Порівняння часу агрегаційної вибірки зі сховища (набір даних H2)

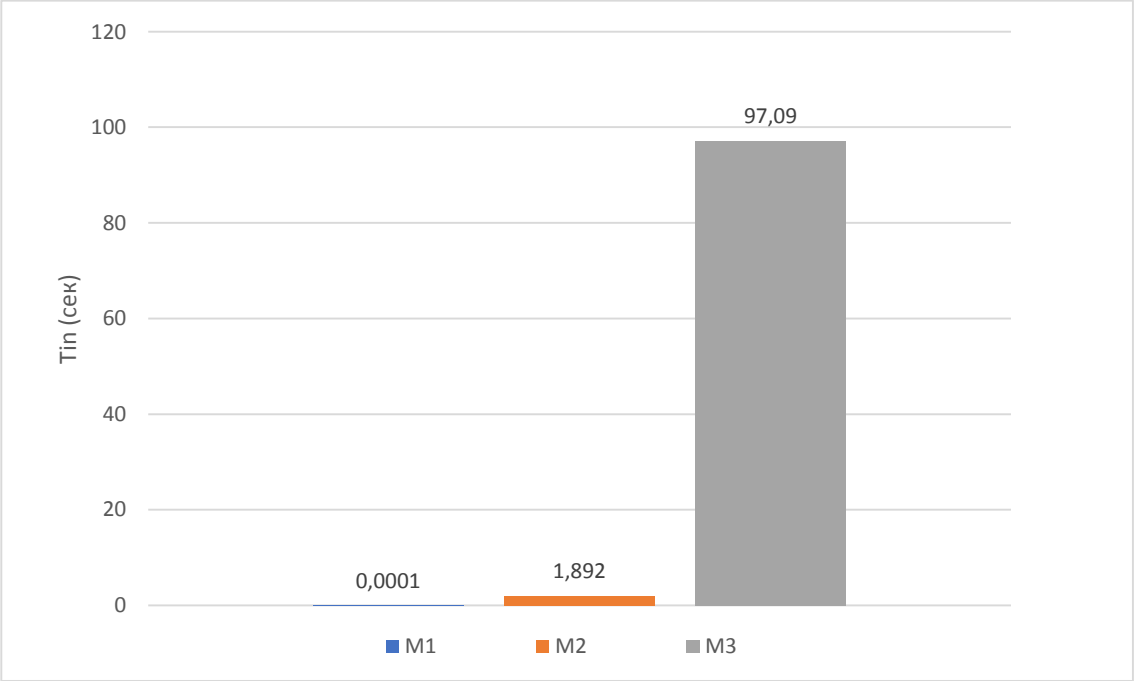


Рисунок 5.12 – Порівняння часу агрегаційної вибірки зі сховища (набір даних H3)

Як видно із вищенаведених графіків та таблиць, запропонований у цій роботі метод програє існуючим методам за вимогами до постійної пам’яті на диску, але виграє у швидкості виконання запитів. Виграш особливо помітний на великих об’ємах даних і сягає тисячократного прискорення.

5.3 Висновок

У цьому розділі проведено експериментальне порівняння методів обробки надвеликих масивів даних у форматі XML. Було виконано ряд експериментів на тестових наборах даних.

Порівняння швидкості виконання запитів із використанням різних методів обробки даних показало, що запропонований алгоритм працює найшвидше з розглянутих нами. Прискорення на великих обсягах даних дає прискорення близько 1000 раз. Варто підмітити, що зі зростанням кількості даних зростає і прискорення, що свідчить про ефективність використання масово паралельних обчислень.

Недоліком запропонованого алгоритму є те, що він використовує середньому втричі більше пам'яті для обробки однакового обсягу даних порівняно із іншими методами.

ВИСНОВОК

У даній магістерський дисертації було розроблено та реалізовано математичне та програмне забезпечення для обробки надвеликих масивів даних у форматі XML. Програмна реалізація була створена за допомогою високорівневої мови програмування .NET, а також компонентів ElasticSearch та Kibana на базі ОС Windows, що дозволяє використовувати середовище розробки Microsoft Visual Studio та засіб для розгортання Docker.

На першому етапі дослідження було проаналізовано природу XML документів, розглянуто XML як деревоподібну структуру даних, досіджено способи зберігання XML документів. Розглянуто альтернативні формати зберігання даних, здійснено їх порівняння із форматом XML. Виявлено, що існуючі способи зберігання XML документів не є оптимальними для здійснення запитів до надвеликих масивів таких документів.

Було розглянуто можливі варіанти прискорення обробки надвеликих масивів XML документів. Було виявлено, що існує теоретична можливість оптимізації зберігання XML документів з попередньою індексацією, а також застосування масового паралельного виконання запитів для збільшення швидкості обробки документів. На основі теоретичних досліджень було ствоерно метод оптимізації обробки надвелиики масивів даних у форматі XML.

Було створене прогамне забезпечення, що реалізує розроблений метод. За його допомогою було експериментально продемонстровано ефективність нового методу. Результати експериментального дослідження показали, що новий метод істотньою перевершує існуючі методи за швидкістю оброблення запитів до надвеликих масивів документів XML.

Наукова новизна отриманих результатів полягає в тому, що вперше було розроблено ефективний спосіб обробки надвеликих масивів XML даних із застосуванням масового паралельного виконання запитів.

Таким чином усі поставленні завдання даної роботи були виконанні і була досягнута поставлена мета. Було створено ефективний метод обробки надвеликих масивів даних у форматі XML.

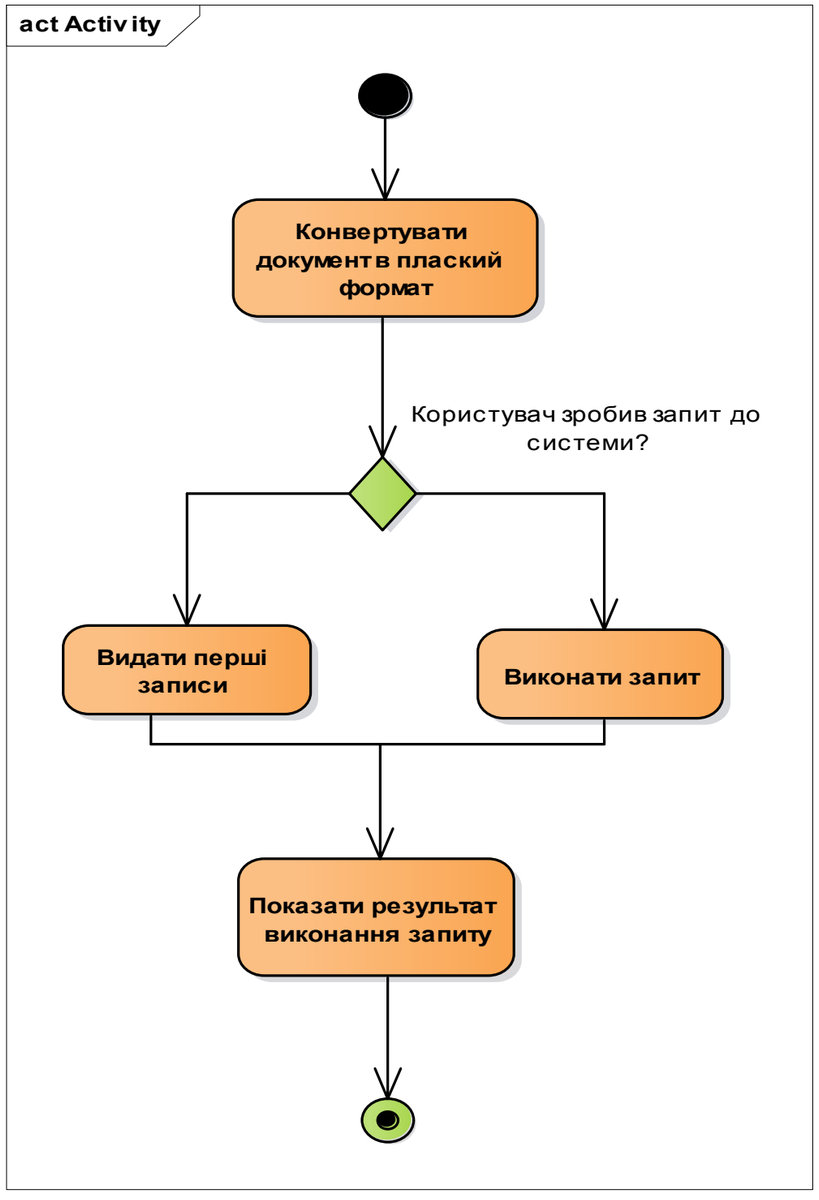
ПЕРЕЛІК ПОСИЛАНЬ

1. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Електронний ресурс] / T.Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler. – 2008. – Режим доступу до ресурсу: https://www.w3.org/TR/xml/
2. XML Tutorial [Електронний ресурс] // W3CSchools – Режим доступу до ресурсу: https://www.w3schools.com/xml/
3. Williams I. Beginning XSLT and XPath: Transforming XML Documents and Data / Ian Williams. – Indianapolis: Wiley Publishing, 2009. – 397 с.
4. Tian F. et al. The design and performance evaluation of alternative XML storage strategies //ACM Sigmod Record. – 2002. – Т. 31. – №. 1. – С. 5-10.
5. Abiteboul S., Cluet S., Milo T. Querying and Updating the File //VLDB. – 1993. – Т. 93. – №. 19. – С. 73-84.
6. Florescu D., Kossmann D. Storing and querying XML data using an RDMBS //IEEE data engineering bulletin. – 1999. – Т. 22. – С. 3.
7. Kanne C. C., Moerkotte G. Efficient storage of XML data //In Proceedings of the 16th International Conference on Data Engineering. – 2000.
8. J. Shanmugasundaram, K. Tufte, et al. Relational Databases for Querying XML Documents: Limitations and Opportunities. VLDB 1999.
9. Excelon [Електронний ресурс] – Режим доступу до ресурсу: http://www.odi.com/excelon.
10. POET [Електронний ресурс] – Режим доступу до ресурсу: http://www.poet.com/
11. Oracle Database [Електронний ресурс] – Режим доступу до ресурсу: www.oracle.com/database/
12. What you'll love about SQL Server 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/en-US/sql-server/sql-server-2019
13. MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/

14. Comer D. Ubiquitous B-tree //ACM Computing Surveys (CSUR). – 1979. – Т. 11. – №. 2. – С. 121-137.
15. Melton J., Buxton S. Querying XML: XQuery, XPath, and SQL/XML in context. – Morgan Kaufmann, 2011.
16. Oracle Berkeley DB XML [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/cd/E17276_01/html/toc.htm
17. Nicola M., John J. Xml parsing: a threat to database performance //Proceedings of the twelfth international conference on Information and knowledge management. – ACM, 2003. – С. 175-178.
18. Chandar J. Join algorithms using map/reduce //Magisterarb. University of Edinburgh. – 2010.
19. J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1):107–113, 2008.
20. A Comparison Of Serialization Formats [Електронний ресурс] – Режим доступу до ресурсу: https://blog.mbedded.ninja/programming/serialization-formats/a-comparison-of-serialization-formats/
21. XMLData Repository [Електронний ресурс] – Режим доступу до ресурсу: http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/
22. Основы технологий XML - СПб: НИУ ИТМО, 2013. – 56 с.
23. Створення та обробка баз даних / Л. С.Глоба, М. Ю. Терновой, Р. Л. Новогрудська, О. С. Штогрина. – Київ, 2013. – 477 с. – (Київський національний технічний університет "Київський політенхнічний інститут").
24. Elasticsearch introduction [Електронний ресурс] // Elasticsearch B.V.. – 2019. – Режим доступу до ресурсу: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html.

25. Clinton Gormley. Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine / Clinton Gormley, Zachary Tong. – Cambridge: O'Reilly Media, Inc, 2015. – 724 с.
26. Introduction to Kibana [Електронний ресурс] // Elasticsearch B.V.. – 2019. – Режим доступу до ресурсу: www.elastic.co/guide/en/kibana/current/introduction.html.
27. Педоренко О. Р. Метод обробки надвеликих масивів XML даних / О. Р. Педоренко, Ю. О. Олійник. // 3 всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019). – 2019.

Додаток А Схема структурна діяльності обробки надвеликих



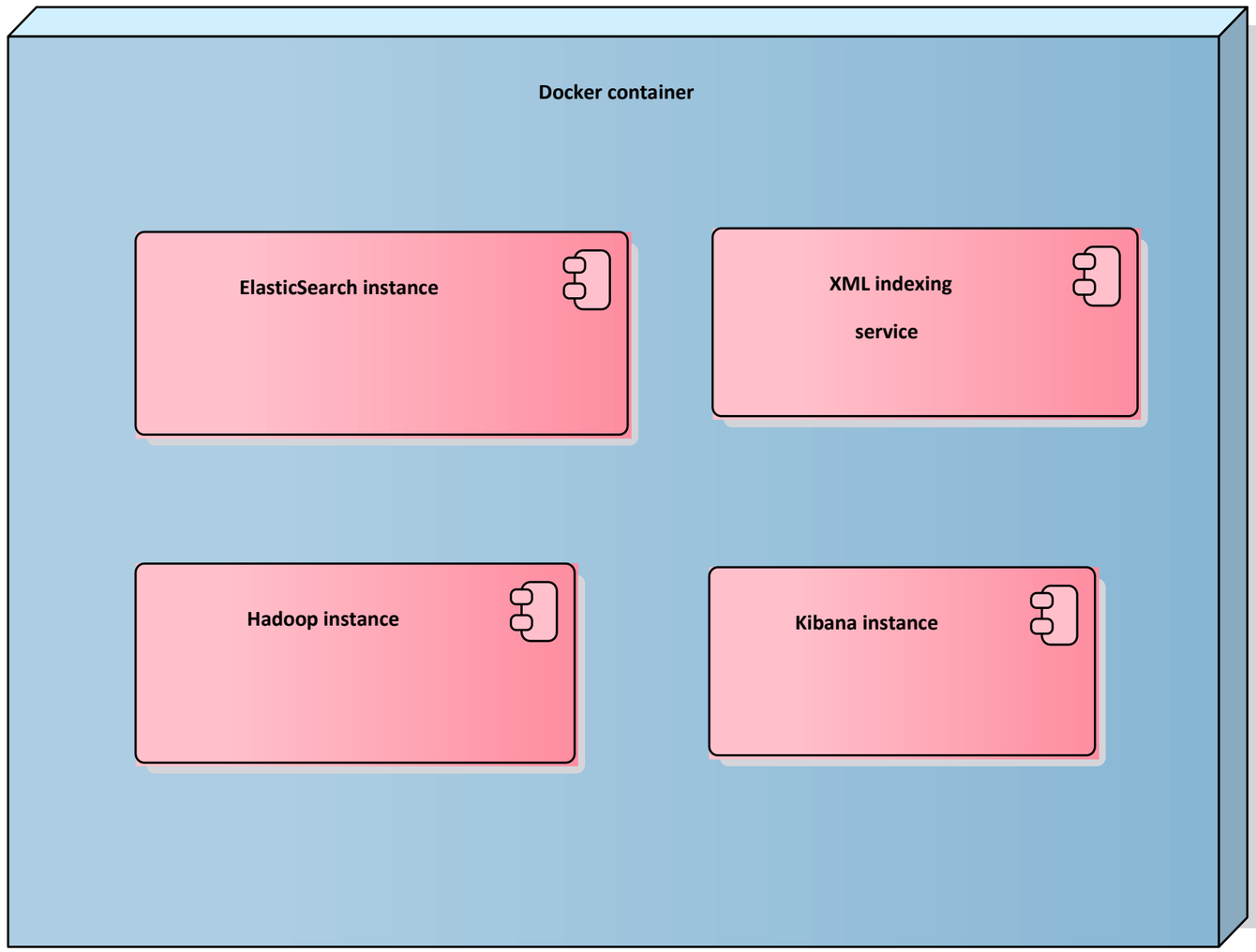
Демонстраційний плакат до магістерської дисертації

Схема структурна діяльності обробки надвеликих масивів XML даних

Виконав(ла) студент(ка) гр. ІП-81мп Педоренко О. Р.

Керівник Олійник Ю. О.

Додаток Б Схема структурна компонентів програмного забезпечення



Демонстраційний плакат до магістерської дисертації

Схема структурна компонентів програмного забезпечення

Виконав(ла) студент(ка) гр. ІП-81мп — Педоренко О. Р.

Керівник — Олійник Ю. О.